A New Necessary Condition for Threshold Function Identification

Chia-Chun Lin, Chin-Heng Liu, Yung-Chih Chen, and Chun-Yao Wang, *Member, IEEE*

***Abstract*—This article proposes a new necessary condition and the corresponding speedup strategies to the threshold function (TF) identification problem. The state-of-the-art to this identification problem could be very time-consuming when the function-under-identification is a non-TF with the unateness property. The proposed new necessary condition can be seamlessly integrated into this identification algorithm. As compared with the state-of-the-art, the improved identification algorithm with the proposed necessary condition can more effectively and efficiently detect non-TFs. Furthermore, according to the experimental results, the ratio of CPU time overhead in the process of checking the proposed necessary condition for identifying all the 8-input TF is only 0.1%.**

***Index Terms*—Digital circuit, linear threshold logic gate, SAT solver, threshold function (TF) identification.**

I. Introduction

Traditionally, people use Boolean logic to represent switching functions. However, in addition to Boolean logic, threshold logic can also represent switching functions. Instead of using multiple Boolean logic gates, a single linear threshold gate (LTG), as the one in Fig. 1(a), can represent a complex function. The elements of an LTG are n binary inputs $x_1, x_2, \ldots, x_n$ with weights $w_1, w_2, \ldots, w_n$, and a threshold value T. The output f is 1 if the summation of each product $x_i \cdot w_i$ is greater than or equal to the threshold value T. Otherwise, the output f is 0. The functionality of an LTG could be changed when we modify any of these parameters, as shown in Fig. 1(b) and (c). Therefore, there are different functions, but not all, that can be represented by a single LTG. A function that can be represented by an LTG is called threshold function (TF). Identification of a TF and then representing it in its canonical form facilitate the succeeding logic synthesis and equivalence checking problems [1], [3], [8], [9].

There were some studies focusing on the TF identification and threshold logic network synthesis in [2]–[6] and [10]–[12]. For example, Neutzling *et al.* [10] proposed an algorithm to assign the weights and threshold value of a TF. The authors listed the inequality system from the irredundant sum-of-products (ISOPs) form of a function, and searched for the weights and threshold value without violating the consistency of the inequality system. Once an assignment of weights and threshold value is obtained, the function is identified as a TF, and the assignment represents the weights and threshold value of the corresponding LTG; otherwise, the function is still *undetermined*.

Recently, Liu *et al.* [4] founded that some TFs cannot be correctly identified by the work of [10]. This is because the *flip situation* occurred in the weight assignment procedure for these TFs. The flip situation could cause the weight assignment procedure in the work of [10] to be inefficient or even failed. Hence, Liu *et al.* [4] improved the algorithm by first simplifying the system of inequalities. Then, they proposed a new weight assignment procedure that incrementally searches a feasible solution for the system of inequalities. This procedure is similar to the breadth-first search (BFS) algorithm, traveling along with all the possibilities of the paths. Liu *et al.* [4] also proposed several strategies to bound the searched paths without traversing the whole solution space. Fig. 2 is an example showing the new weight assignment procedure in the work of [4]. In this new weight assignment procedure, however, if a feasible assignment cannot be found when the weights reach their theoretical upper bound values, the function will be identified as an undetermined function. Although the new weight assignment procedure in [4] solved the problem of flip situation and identified all the TFs with eight inputs, the procedure could be time-consuming when the function-under-identification is a non-TF with the unateness property. Both the works in [4] and [10] used the property of unateness to accelerate the process of TF identification because unateness is a necessary condition for being TFs. Once the non-TFs have the unateness property, the TF identification flow in [4] cannot remove the function-under-identification in the preprocessing stage. As a result, the succeeding weight assignment procedure will continue until one weight reaches its theoretical upper bound. An example illustrating this weight assignment procedure for non-TFs is shown in Fig. 3. We can see that the procedure searches the weights incrementally and iteratively. According to a table in the work of [7], which is Table I in this article, we find that the number of NP-equivalence classes of unate functions with five variables is 16 143. However, the number of NP-TFs of five variables is only 92. This indicates that many unate functions are non-TFs. Passing unateness check as a criteria to run the weight assignment procedure is not appropriate though. Thus, to improve the TF identification algorithm, a more efficient method for screening out non-TFs is desired. As a result, in this article, in addition to the known unateness property, we propose a new necessary condition for a function being a TF, and use it to identify non-TFs earlier in the improved TF identification algorithm.

II. Preliminaries

A. Hyperplane and Half-Space

Hyperplane and half-space have close connections with an LTG, and they play important roles in the succeeding discussion. Therefore, in this section, let us explain the relationship among the hyperplane, half-space, and LTG first.

An LTG can be represented as a hyperplane H in an n-dimensional space, i.e., $H : \sum_{i=1}^{n} x_i w_i = T$. The half-spaces described by

Manuscript received June 17, 2019; revised August 30, 2019 and October 28, 2019; accepted December 28, 2019. Date of publication January 8, 2020; date of current version November 20, 2020. This work was supported by the Ministry of Science and Technology of Taiwan under Grant MOST 107-2221-E-155-046, Grant MOST 108-2221-E-155-047, Grant MOST 106-2221-E-007-111-MY3, and Grant MOST 108-2218-E-007-061. This article was recommended by Associate Editor Z. Zhang. *(Corresponding author: Chia-Chun Lin.)*

Chia-Chun Lin, Chin-Heng Liu, and Chun-Yao Wang are with the Department of Computer Science, National Tsing Hua University, Hsinchu 30013, Taiwan (e-mail: chiachunlin@gapp.nthu.edu.tw; posada2968@yahoo.com.tw; wcyao@cs.nthu.edu.tw).

Yung-Chih Chen is with the Department of Computer Science and Engineering, Yuan Ze University, Taoyuan 32003, Taiwan (e-mail: ycchen.cse@saturn.yzu.edu.tw).

Digital Object Identifier 10.1109/TCAD.2020.2964768

TABLE I
NUMBER OF n-INPUT FUNCTIONS IN DIFFERENT CLASSES [7]

n	2	3	4	5	6	7	8
Boolean functions of n variables	10	218	64,594	$\approx 4.3 \times 10^9$	$\approx 1.8 \times 10^{19}$	$\approx 3.4 \times 10^{38}$	$\approx 1.16 \times 10^{77}$
NP-equivalence classes of Boolean functions of n variables	3	16	380	1,227,756	400,507,805,615,570	-	-
NP-equivalence classes of unate functions of n variables	10	20	180	16,143	-	-	-
NP-threshold functions of n variables	2	5	17	92	994	28,262	2,700,791

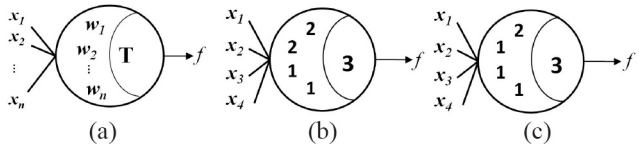

Fig. 1. (a) LTG model. (b) Boolean function $f = x_1x_2 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4$ represented by an LTG. (c) Another Boolean function $f = x_1x_2 + x_1x_3 + x_1x_4 + x_2x_3x_4$ represented by an LTG.

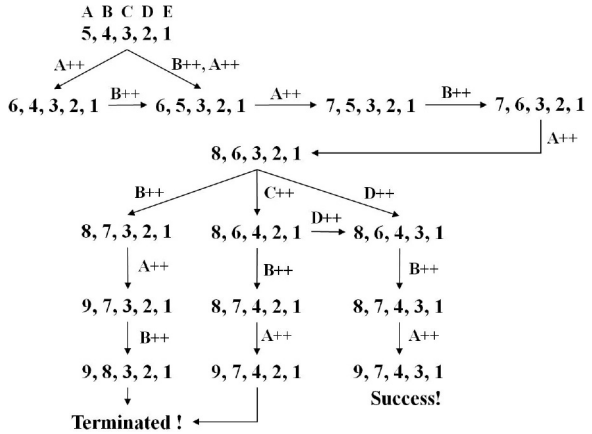

Fig. 2. Weight assignment process for $f = x_1x_2x_3 + x_1x_2x_4 + x_1x_2x_5x_6 + x_1x_2x_5x_7 + x_1x_3x_4 + x_1x_3x_5x_6 + x_1x_4x_5x_6 + x_2x_3x_4 + x_2x_3x_5x_6 + x_2x_4x_5x_6$ from [4]. There are only five variables in a 7-input function because the variable ordering is $x_1 = x_2 > x_3 = x_4 > x_5 > x_6 > x_7$.

$\sum_{i=1}^{n} x_i w_i \geq T$ and $\sum_{i=1}^{n} x_i w_i < T$ are named as the positive half-space H^+ and the negative half-space H^-, respectively. When any point located in H^+ is applied to the LTG, the output is 1; otherwise, the output is 0. This is the linear separativity property of TFs.

B. Chow's Parameter

Chow's parameter $P(f)$ of a function $f(x_1, \ldots, x_n)$ is a vector defined in (1)

$$P(f) = (p_1(f), p_2(f), \ldots, p_n(f); p_0(f)) \tag{1}$$

where $p_i(f), i = 1 \sim n$, is the number of minterms in the on-set of f for which $x_i = 1$, and $p_0(f)$ is the total number of minterms in the on-set of f. In fact, Chow's parameter provides some clues in TF identification. It is clear that the variable x_i affects function f more when $p_i(f)$ is larger than the others. Therefore, the largest weight in an LTG is always associated with the input having the largest $p_i(f)$.

C. Shannon's Expansion

Shannon's expansion decomposes a function $f(x_1, \ldots, x_n)$ in the n-dimension Boolean space B^n as (2)

$$f(x_1, \ldots, x_n) = Ax_ix_j + Bx_ix_j' + Cx_i'x_j + Dx_i'x_j' \tag{2}$$

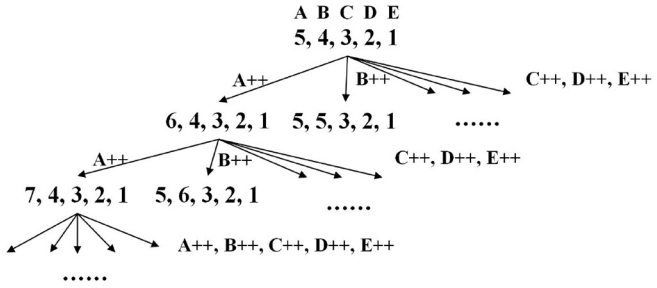

Fig. 3. Weight assignment procedure continues until one of the weights reaches the upper bound when the function is a non-TF.

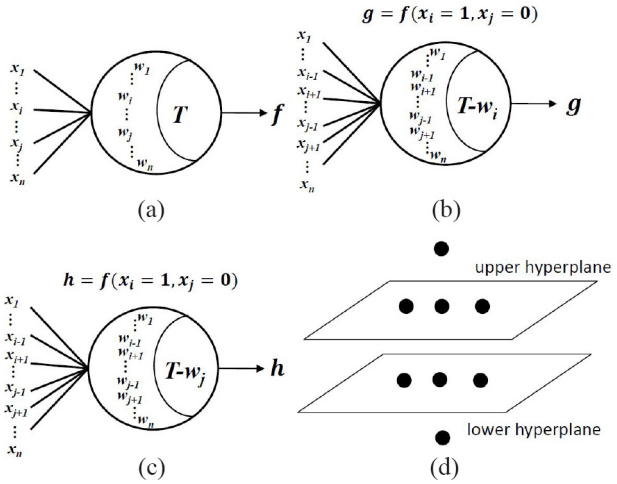

Fig. 4. (a) LTG model for a TF. (b) LTG model for cofactor function $f(x_i = 1, x_j = 0)$. (c) LTG model for cofactor function $f(x_i = 0, x_j = 1)$. (d) Relationship between two hyperplanes corresponding to the LTG in (b) and the LTG in (c).

for all x_i, x_j, where A, B, C, and D are four cofactor functions, respectively. According to the definition of Chow's parameter, we can see that $p_i(f)$ equals the total number of minterms in the on-set of A and B in B^{n-2}. Similarly, $p_j(f)$ equals the total number of minterms in the on-set of A and C in B^{n-2}. Both previous works [4], [10] assigned initial values to the weights in the weight assignment procedure based on Chow's parameter. However, two variables having the same Chow's parameter do not imply that these two variables have the same weight in its LTG. For example, Chow's parameter of function $ab + cd$ is $(5, 5, 5, 5; 7)$. Although variables a and c have the same value in its Chow's parameter, they come from different components in the Shannon's expansion. In fact, a more precise way to figuring out the contributions of two variables to the function is to compare the two cofactor functions $f(x_i = 1, x_j = 0)$ and $f(x_i = 0, x_j = 1)$. More details about this idea will be presented in Section III.

III. Nonthreshold Function Identification

A. New Necessary Condition for Function Being TF

Definition: Two functions f and g have the implication relation if and only if $f \subseteq g$ or $g \subseteq f$.

Theorem 1: If an n-input function f(x) is a TF, then its two cofactor functions, $f(x_i = 1, x_j = 0)$ and $f(x_i = 0, x_j = 1)$, have the implication relation, for all input pairs x_i, x_j.

Proof: Since $f(x)$ is a TF, it can be represented as an LTG by definition. Without loss of generality, we assume that the weights are $w_1, w_2, \ldots, w_n$ with respect to input variables $x_1, x_2, \ldots, x_n$, and the threshold value is T, as shown in Fig. 4(a). To obtain the corresponding LTG for the cofactor function $f(x_i = 1, x_j = 0)$, we remove the inputs x_i, x_j and the weights w_i, w_j from the original LTG, and update the threshold value. Since the input $x_i = 1$ contributes the weight w_i to the weight summation, the threshold value T needs to be updated as $(T - w_i)$ after removing x_i, x_j and w_i, w_j. The resultant LTG of $f(x_i = 1, x_j = 0)$ is shown in Fig. 4(b). Similarly, the resultant LTG of cofactor function $f(x_i = 0, x_j = 1)$, as shown in Fig. 4(c), is obtained by removing x_i, x_j and w_i, w_j, and updating the threshold value as $(T - w_j)$. According to the explanation in Section II-A, we have known that there exists a corresponding hyperplane for any given LTG. The only difference between the LTGs in Fig. 4(b) and (c) is the threshold value. That is, the hyperplanes represented by these two LTGs are parallel to each other, as shown in Fig. 4(d). Since these two hyperplanes with respect to the LTGs of cofactor functions $f(x_i = 1, x_j = 0)$ and $f(x_i = 0, x_j = 1)$ are parallel to each other, the positive half-space of the upper hyperplane will be a subset of the positive half-space of the lower hyperplane. As a result, we can conclude that these two cofactor functions have the implication relation for any input pairs x_i, x_j without knowing the position of each LTG's hyperplane. ■

B. Implication Relation Check by Using SAT Solvers

In Theorem 1, we have known that the satisfaction of implication relation for every pair of input variables is the necessary condition for being a TF. The next issue to be dealt with is how to check whether this relation holds or not efficiently. In this article, we model this problem as a SAT problem and use SAT solvers [13] to obtain the result efficiently.

When two cofactor functions g and h of a function f do not have the implication relation, there exist two input patterns a_1 and a_2 such that $g(a_1) = 1$, $h(a_1) = 0$ and $g(a_2) = 0$, $h(a_2) = 1$. We construct a network, as shown in Fig. 5, and use SAT solvers to check whether the cofactor functions g and h have the implication relation or not. First, the output of the network in Fig. 5 is set to be 1. When the SAT solver returns a satisfiable input pattern for the conjunctive normal form (CNF) of the network, it means that there exists an input pattern a_1 such that $g(a_1) = 1$, $h(a_1) = 0$, and exists an input pattern a_2 such that $g(a_2) = 0$, $h(a_2) = 1$. As a result, the two cofactor functions do not have the implication relation, and f is not a TF. If the SAT solver returns UNSAT, then g and h have the implication relation.

C. MORE Efficient Method

To realize an n-input function-under-identification f as a non-TF in the improved identification algorithm, we have to examine every pair of input variables, which is with the time complexity of $O(n^2)$. In fact, we could obtain the same result more efficiently without checking all of these input pairs under a certain situation. The idea behind this is transitive law. We use Theorem 2 to express this idea.

Theorem 2: Given an n-input function $f(x)$, if $f(x_i = 1, x_j = 0) = f(x_i = 0, x_j = 1)$ and $f(x_j = 1, x_k = 0) = f(x_j = 0, x_k = 1)$, then $f(x_i = 1, x_k = 0) = f(x_i = 0, x_k = 1)$, for all inputs x_i, x_j, x_k.

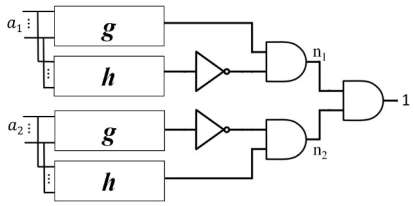

Fig. 5. Model for checking the implication relation of function g and function h.

Proof: The premises are indexed as (3) and (4)

$$f(x_i = 1, x_j = 0) = f(x_i = 0, x_j = 1) \tag{3}$$

$$f(x_j = 1, x_k = 0) = f(x_j = 0, x_k = 1). \tag{4}$$

First, we rewrite (3) as $f(x_i = 1, x_j = 0, x_k = 0) = f(x_i = 0, x_j = 1, x_k = 0)$ under $x_k = 0$. Then, we rewrite (4) as $f(x_i = 0, x_j = 1, x_k = 0) = f(x_i = 0, x_j = 0, x_k = 1)$ under $x_i = 0$. Hence, we have

$$f(x_i = 1, x_j = 0, x_k = 0) = f(x_i = 0, x_j = 0, x_k = 1) \tag{5}$$

by transitive law. Similarly, we rewrite (3) as $f(x_i = 1, x_j = 0, x_k = 1) = f(x_i = 0, x_j = 1, x_k = 1)$ under $x_k = 1$, and rewrite (4) as $f(x_i = 1, x_j = 1, x_k = 0) = f(x_i = 1, x_j = 0, x_k = 1)$ under $x_i = 1$. Hence, we have

$$f(x_i = 1, x_j = 1, x_k = 0) = f(x_i = 0, x_j = 1, x_k = 1) \tag{6}$$

by transitive law again. As a result, by examining (5) and (6), no matter what the value of x_j is, $f(x_i = 1, x_k = 0) = f(x_i = 0, x_k = 1)$. ■

By Theorem 1, we have known that a function-under-identification is a non-TF if a pair of input variables whose cofactor functions do not have the implication relation was found. Theorem 2 states that some redundant computations can be skipped when certain cofactor functions are equivalent. Theorem 3 then connects Theorem 1 with Theorem 2, and is used to facilitate the detection of the proposed necessary condition.

Theorem 3: Given two variables x_i and x_j in a function $f(x_1, , \ldots, x_n)$ with their corresponding Chow's parameter $p_i(f)$ and $p_j(f)$, if $p_i(f)$ equals $p_j(f)$, then these two cofactor functions $f(x_i = 1, x_j = 0)$ and $f(x_i = 0, x_j = 1)$ either are equivalent or do not have the implication relation.

Proof: Here, we use the contraposition to prove this theorem. First, we rewrite this theorem as the following statement:

$$P \rightarrow (Q_1\bar{Q}_2 + \bar{Q}_1Q_2) \tag{7}$$

where P, Q_1, and Q_2 represent $p_i(f)$ equals $p_j(f)$, $f(x_i = 1, x_j = 0)$ and $f(x_i = 0, x_j = 1)$ are equivalent, and $f(x_i = 1, x_j = 0)$ and $f(x_i = 0, x_j = 1)$ do not have the implication relation, respectively. Equation (7) can be rewritten as (8) by contraposition

$$(Q_1Q_2 + \bar{Q}_1\bar{Q}_2) \rightarrow \bar{P} \tag{8}$$

where $Q_1Q_2 + \bar{Q}_1\bar{Q}_2 = \overline{Q_1\bar{Q}_2 + \bar{Q}_1Q_2}$. However, in (8), Q_1Q_2 means that two cofactor functions $f(x_i = 1, x_j = 0)$ and $f(x_i = 0, x_j = 1)$ are equivalent, but do not have the implication relation, which conflicts with itself. Therefore, we have $Q_1Q_2 = 0$ and (8) can be simplified as (9)

$$(\bar{Q}_1\bar{Q}_2) \rightarrow \bar{P}. \tag{9}$$

Hence, in the following paragraphs, we only need to prove (9) for this theorem. That is, if the two cofactor functions are not equivalent, and they have the implication relation, the corresponding Chow's parameter $p_i(f) \neq p_j(f)$. Note that when two cofactor functions are not

TABLE II
THEORETICAL WEIGHT UPPER BOUND FOR n-INPUT TFS [7]

n	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Theoretical weight upper bound	1	2	3	6	14	32	76	195	521	1,458	4,248	12,867	40,389	131,072

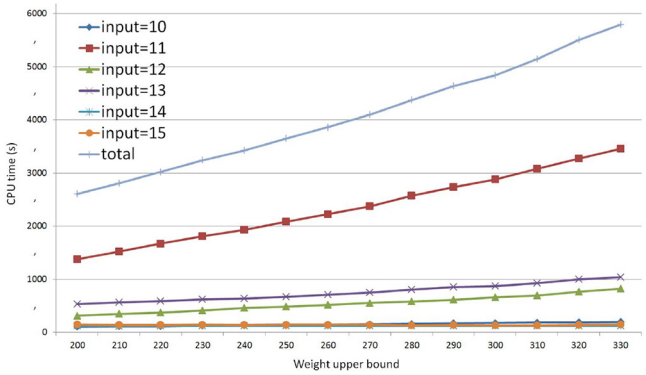

Fig. 6. Required CPU time for identifying all the 600 000 non-TFs under different weight upper bounds in [4].

TABLE III
CPU TIME COMPARISON FOR IDENTIFYING ALL THE 600 000 NON-TFS BETWEEN [4] AND OUR APPROACH

|Input|	[4]	Ours	
		w/o Thms. 2 and 3 (s)	w/ Thms. 2 and 3 (s)
10	199.28	19.78	5.14
11	3455.60	22.46	5.34
12	823.17	24.87	6.25
13	1042.23	27.63	7.01
14	124.21	33.11	6.17
15	149.47	39.73	6.29
Total	5793.96	167.58	36.20
Speedup	1	34	160

equivalent, and have the implication relation, one cofactor function must be a *proper subset* of the other cofactor function.

According to Shannon's expansion in (2) of Section II-E, we know that $p_i(f)$ equals the total number of minterms in the on-set of A and B in B^{n-2}, and $p_j(f)$ equals the total number of minterms in the on-set of A and C in B^{n-2}. Similarly, the number of minterms in the cofactor function $f(x_i = 1, x_j = 0)$ equals the number of minterms in the on-set of B in B^{n-2}, and the number of minterms in the cofactor function $f(x_i = 0, x_j = 1)$ equals the number of minterms in the on-set of C in B^{n-2}. Since given that one cofactor function is a proper subset of the other cofactor function, the number of minterms in the on-set of these two cofactor functions are different. In other words, the number of minterms in the on-set of B in B^{n-2} is not equal to the number of minterms in the on-set of C in B^{n-2}. Next, we add the minterms in the on-set of A in B^{n-2} to the on-sets of B, and C in B^{n-2}. As a result, the total number of minterms in the on-set of A and B in B^{n-2} is not equal to that of A and C in B^{n-2}, i.e., the corresponding Chow's parameter $p_i(f) \neq p_j(f)$. ■

IV. EXPERIMENTAL RESULTS

We implemented the proposed algorithm in C++ language. The experiments were conducted on a 2.6-GHz Linux platform (CentOS 6.7). Since the proposed necessary condition focuses on the identification of non-TFs, we randomly generate 100 000 unate but non-TFs for each group of functions with the same number of inputs from 10 to 15 inputs as the benchmarks. The total amount of benchmarks is 600 000. We compare the CPU time for identifying these functions with the state-of-the-art [4].

In the identification algorithm of the state-of-the-art, a theoretical weight upper bound is needed for setting a termination condition in the weight assignment procedure. If the assigned largest weight is greater than this theoretical weight upper bound, the function-under-identification is identified as an undetermined function. Table II shows the theoretical weight upper bounds for n-input TFs. However, since these theoretical upper bound values are quite large, we only set them from 200 to 330 for showing the trend of the CPU time growth in the work of [4]. Besides, we conducted the experiments on the functions with no more than 15 inputs because the previous work of [4] only considered these TFs in the experiments. In fact, the proposed theorems can be applied to the functions with more than 15 inputs without significant overheads. Fig. 6 shows the required CPU time in the identification algorithm of [4] under these different upper bounds. In these experiments, all the 600 000 non-TFs are identified as undetermined functions. For achieving this, [4] spent 5793.96 s in total under the largest weight upper bound of 330. In Fig. 6, we can see that the required CPU time grows significantly with the increase of this upper bound. Besides, we know that the required CPU time is still underestimated since we only set the largest upper bound as 330.

On the other hand, to show the effectiveness and efficiency of the proposed necessary condition, we also conduct the experiments for identifying these 600 000 functions using the improved identification algorithm. All of these functions are successfully identified as non-TFs rather than undetermined functions. The required CPU time is shown in Table III. In Table III, we also show the performance comparison with/without applying Theorems 2 and 3 in the algorithm. Column 1 lists the number of inputs. Column 2 shows the required CPU time for identifying these functions by [4] with the weight upper bound of 330. Column 3 shows the required CPU time for identifying these functions without applying Theorems 2 and 3. Column 4 shows the corresponding required CPU time with applying Theorems 2 and 3 further. According to Table III, [4] spent 5793.96 s to identify all these 600 000 functions as undetermined functions. The proposed improved algorithm spent 167.58 s to identify all these functions as non-TFs when only applying Theorem 1. However, the required CPU time is further reduced to 36.2 s when Theorems 2 and 3 are also applied. These results demonstrate that Theorems 2 and 3 elevate the performance of the improved identification algorithm significantly. In summary, the speedup of the improved identification algorithm reaches around 160× as compared with the state-of-the-art [4].

Note that since the 600 000 non-TFs in the experiments of Table III were generated randomly, it is possible that a generated non-TF encountered the worst-case situation in the weight assignment procedure of [4], which is shown in Fig. 3. In our experiments, the non-TFs in the generated 11-input functions encountered this situation more. This is the reason why the CPU time for this group of functions is larger. This is also a good example to demonstrate that the CPU time bottleneck in this experiment is the occurrence of the worst case like Fig. 3, rather than the number of inputs in a group.

TABLE IV
CPU TIME OVERHEAD IN THE IMPROVED IDENTIFICATION ALGORITHM FOR 1 TO 8-INPUT TFS

|Input|	CPU time (s) [4]	Overhead (s)	Ratio (%)
1	<0.01	<0.01	-
2	<0.01	<0.01	-
3	<0.01	<0.01	-
4	<0.01	<0.01	-
5	<0.01	<0.01	-
6	<0.01	<0.01	-
7	8.78	1.50	17.10
8	251,700	321	0.10

Although the improved TF identification algorithm can remove non-TFs earlier, it may introduce CPU time overhead for the functions-under-identification that are TFs indeed. In fact, a TF needs to run the necessary condition checking for much more times than a non-TF does. This is because every pair of cofactor functions needs to be examined when the function is a TF. For a non-TF, however, if we found that one pair of cofactor functions do not have the implication relation, the checking process can be terminated immediately. Therefore, in the experiments of the second part, we would like to show the CPU time overhead in the improved identification algorithm when identifying all the TFs with 1 to 8 inputs. The experimental results are shown in Table IV. Column 1 lists the number of inputs. Column 2 lists the CPU time for identifying all the n-input TFs in [4]. Column 3 is the CPU time overhead in the improved identification algorithm. Column 4 shows the ratio of CPU time overhead. According to Table IV, we can see that the ratio of CPU time overhead for identifying all the 8-input TFs is only 0.1%. In other words, the proposed necessary condition for removing non-TFs will not incur much CPU time overhead to the original identification algorithm for TFs. In summary, the proposed ideas in this article complete the identification algorithm for both TFs and non-TFs.

V. CONCLUSION

The state-of-the-art does not well deal with the TF identification problem when the function-under-identification is a non-TF. Hence, in this article, we propose a new necessary condition for a function being a TF. The checking process for this necessary condition is also modeled and explained in detail. The experimental results show that the contributions of this article complete the identification algorithm for both TFs and non-TFs.

REFERENCES

[1] Y.-C. Chen, R. Wang, and Y.-P. Chang, "Fast synthesis of threshold logic networks with optimization," in *Proc. 21st Asia South Pac. Design Autom. Conf. (ASP-DAC)*, 2016, pp. 486–491.
[2] P.-Y. Kuo, C.-Y. Wang, and C.-Y. Huang, "On rewiring and simplification for canonicity in threshold logic circuits," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, 2011, pp. 396–403.
[3] S.-Y. Lee, N.-Z. Lee, and J.-H. R. Jiang, "Canonicalization of threshold logic representation and its applications," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, San Diego, CA, USA, 2018, pp. 1–8.
[4] C.-H. Liu, C.-C. Lin, Y.-C. Chen, C.-C. Wu, C.-Y. Wang, and S. Yamashita, "Threshold function identification by redundancy removal and comprehensive weight assignments," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 12, pp. 2284–2297, Dec. 2019.
[5] C.-C. Lin, C.-Y. Wang, Y.-C. Chen, and C.-Y. Huang, "Rewiring for threshold logic circuit minimization," in *Proc. Design Autom. Test Europe Conf. Exhibit. (DATE)*, Dresden, Germany, 2014, pp. 1–6.
[6] C.-C. Lin, C.-W. Huang, C.-Y. Wang, and Y.-C. Chen, "In&Out: Restructuring for threshold logic network optimization," in *Proc. 18th Int. Symp. Qual. Electron. Design (ISQED)*, 2017, pp. 413–418.
[7] S. Muroga, *Threshold Logic and its Applications*. New York, NY, USA: Wiley, 1971.
[8] A. Neutzling, J. M. Matos, A. I. Reis, R. P. Ribas, and A. Mishchenko, "Threshold logic synthesis based on cut pruning," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Austin, TX, USA, 2015, pp. 494–499.
[9] A. Neutzling, J. M. Matos, A. Mishchenko, A. I. Reis, and R. P. Ribas, "Effective logic synthesis for threshold logic circuit design," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 5, pp. 926–937, May 2019.
[10] A. Neutzling, M. G. A. Martins, V. Callegaro, A. I. Reis, and R. P. Ribas, "A simple and effective heuristic method for threshold logic identification," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 5, pp. 1023–1036, May 2018.
[11] R. O. Winder, "Enumeration of seven-argument threshold functions," *IEEE Trans. Electron. Comput.*, vol. EC-14, no. 3, pp. 315–325, Jun. 1965.
[12] R. Zhang, P. Gupta, L. Zhong, and N. K. Jha, "Threshold network synthesis and optimization and its application to nanotechnologies," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 24, no. 1, pp. 107–118, Jan. 2005.
[13] *MiniSAT*. Accessed: Oct. 28, 2019. [Online]. Available: http://minisat.se/