Analytic Approaches to the Collapse Operation and Equivalence Verification of Threshold Logic Circuits

Nian-Ze Lee, Hao-Yuan Kuo, Yi-Hsiang Lai and Jie-Hong Roland Jiang
Graduate Institute of Electronics Engineering / Department of Electrical Engineering
National Taiwan University, Taipei, 10617, Taiwan

ABSTRACT

Threshold logic circuits gain increasing attention due to their feasible realization with emerging technologies and their strong bind to neural network applications. In this paper, for logic synthesis we formulate the fundamental operation of collapsing threshold logic gates, not addressed by prior efforts. A necessary and sufficient condition of collapsibility is obtained for linear combination of two threshold logic gates, and an analytic approach is proposed for fast circuit transformation. On the other hand, for equivalence verification we propose a linear time translation from threshold logic circuits to pseudo-Boolean constraints, in contrast to prior exponential translation costs. Experimental results demonstrate the effectiveness of circuit transformation by the collapse operation and the memory efficiency of equivalence verification by our pseudo-Boolean translation.

1. INTRODUCTION

While the continuation of Moore's law slows down, different computation paradigms have been considered as possible substitutions for CMOS technologies. Among these possibilities, threshold logic (TL) circuits gain increasing attention due to their strong bind to neural network applications [8, 12] and feasible realization with emerging technologies, such as spintronics, memristors, resonant tunneling devices (RTD), quantum cellular automata (QCA), and single electron transistors (SET) [2, 5, 16].

As the technologies realizing threshold logic become more viable than before, the synthesis and verification of TL circuits are important topics of research to support large scale system construction. Among prior synthesis endeavors, [22] splits a Boolean network into unate nodes and applies linear programming to derive corresponding weights and thresholds; [20] proposes a decomposition algorithm that works directly on truth tables and applies a binate splitting heuristic; a tree matching method is used in [6] to synthesize TL circuits; in [18], implicant-implicit algorithms are proposed to improve synthesis performance; [15] starts from the and-inverter-graph (AIG) and utilizes the well-developed technology mapper in [1] to map the AIG with cuts that are threshold functions. Other efforts on fast identification of TL functions [17], rewiring [11] TL circuits, merging TL gates [3], among others, have also been investigated.

For formal verification of threshold logic circuits, prior work [7] and [23] proposed translation techniques from threshold logic gates to binary decision diagrams (BDD) and conjunctive normal form (CNF) formulas, respectively. They suffer from translation complexity exponential in the fanin size of a threshold logic gate and are not scalable to large designs.

In this paper, the fundamental operation of collapsing (or composing) two threshold logic gates is formulated, which subsumes the restricted merging operations in [3]. Building upon a necessary and sufficient condition of linear composition, an analytic approach is proposed for fast circuit transformation. The collapse (or composition) operation may be combined with other existing techniques to form useful scripts for threshold logic optimization. Moreover, a linear time translation from threshold logic circuits to pseudo Boolean (PB) constraints amenable for formal verification on threshold logic circuits is proposed. Unlike most previous synthesis and verification methods impose restriction on the fanin size of a threshold logic gate, our proposed collapse operation and circuit-to-PB-constraint conversion are not limited to threshold gate sizes. Experiments show promising benefits of our proposed methods.

Collapsing threshold logic gates may potentially result in circuits with high-fanin gates, whose practicality might seem unclear. However we mention a key application to justify the usefulness of collapse operation. In machine learning and brain emulation applications, a neuron in a neural network typically has a large number of fanins. For example, a neuron of a deep convolutional neural network for image classification may have 2,048 inputs [10]. It is therefore common for neuromorphic chips to support high-fanin neurons, e.g., the IBM TrueNorth chip [13] implements programmable neurons with up to 256 inputs and may serve as a feasible platform to realize threshold gates with large fanin sizes. As prior work on threshold logic synthesis mostly considers threshold gates up to a few fanins, e.g., 8 inputs in [15], the collapse operation may help relax the fanin-size restriction for circuits to be mapped into neuromorphic architectures. In addition, it may facilitate the integration of neural networks and control logic circuits into a single neuromorphic chip. On the other hand, the collapse (namely, composition) operation can be used as an elementary command, dual to

ICCAD '16, November 07-10, 2016, Austin, TX, USA

 ISBN 978-1-4503-4466-1/16/11…$15.00

DOI: http://dx.doi.org/10.1145/2966986.2967001

the decomposition operation, in a synthesis script to transform threshold logic circuits for optimization, similar to its Boolean counterpart used in conventional logic synthesis, such as command `eliminate` in `SIS` [19].

2. PRELIMINARIES

A *threshold function* $f : \mathbb{B}^n \to \mathbb{B}$ over variables $(x_1, \ldots, x_n)$ specified by a vector of constant weights $(w_1, \ldots, w_n) \in \mathbb{Z}^n$ and a constant threshold value $T \in \mathbb{Z}$ is a Boolean function that satisfies

$$f(x_1, \ldots, x_n) = \begin{cases} 1, & \text{if } \sum_{i=1}^{n} w_i x_i \geq T, \\ 0, & \text{otherwise.} \end{cases}$$

A *threshold logic gate* (TLG) v with n inputs $(x_1, \ldots, x_n)$ and one output z_v is a logic unit that realizes some threshold function $f_v(x_1, \ldots, x_n)$ with weights $(w_1, \ldots, w_n)$ and threshold value T_v. The valuation of its output variable z_v is determined by $f_v(x_1, \ldots, x_n)$. In the sequel, a threshold logic gate v with weights $(w_1, \ldots, w_n)$ and threshold T_v is denoted as $v = [w_1, \ldots, w_n; T_v]$.

A *threshold logic circuit* (abbreviated *TL circuit*, or *TLC*) $G = (V, E)$ is a directed acyclic graph (DAG), where V is a set of vertices and $E \subseteq V \times V$ is a set of edges. An edge $e = (u, v)$ signifies that vertex v refers to vertex u as an input; u is called a *fanin* of v; v is called a *fanout* of u. V_I (resp. V_O) is a nonempty subset of V such that every vertex in V_I (resp. V_O) has no fanins (resp. fanouts). We assume V_I and V_O are disjoint. A vertex $v \in V_I$ (resp. V_O) is referred to as a *primary input* (PI) (resp. *primary output* (PO)). A vertex $v \in V \backslash V_I$ represents a threshold logic gate. In the sequel, we shall not distinguish between a vertex and its corresponding threshold logic gate.

3. COLLAPSING THRESHOLD LOGIC

In this section, we formulate the collapse operation on TL circuits. We derive the feasibility conditions of collapsing in threshold logic. The collapse operation can be seen as the opposite operation of decomposition, which decomposes a TLG into a composite of multiple TLGs.

3.1 Problem Formulation

Problem Formulation 1 (Collapsing in General). *Given a threshold logic circuit $G(V, E)$, let vertices $u, v \in V$ with $(u, v) \in E$ and assume $u = [a_1, \ldots, a_n; T_u]$ with fanins $(x_1, \ldots, x_n)$ and $v = [b_1, \ldots, b_m; T_v]$ with fanins $(y_1, \ldots, y_m)$. Let $y_1 = z_u$. The* TLG collapsing problem of u to v *asks whether there exists a TLG $\tilde{v} = [c_1, \ldots, c_{n+m-1}; T_{\tilde{v}}]$ with fanins $(x_1, \ldots, x_n, y_2, \ldots, y_m)$ such that $f_{\tilde{v}}(x_1, \ldots, x_n, y_2, \ldots, y_m) = f_v(f_u(x_1, \ldots, x_n), y_2, \ldots, y_m)$ for all assignments to variables $(x_1, \ldots, x_n, y_2, \ldots, y_m)$.*

We remark that the general TLG collapsing problem formulated above involves $n+m$ parameters $c_1, \ldots, c_{n+m-1}, T_{\tilde{v}} \in \mathbb{Z}$ to search for legitimate $\tilde{v}$. The large search space $\mathbb{Z}^{n+m}$ imposes expensive computation. To overcome this obstacle, we consider the collapsing of u to v in the special form of a linear combination of u and v as the following formulation states.

Problem Formulation 2. (Collapsing via Linear Combination). *Given a threshold logic circuit $G(V, E)$, let vertices $u, v \in V$ with $(u, v) \in E$ and assume $u = [a_1, \ldots, a_n;$ $T_u]$ with fanins $(x_1, \ldots, x_n)$ and $v = [b_1, \ldots, b_m; T_v]$ with fanins $(y_1, \ldots, y_m)$. Let $y_1 = z_u$. The* TLG collapsing problem of u to v via linear combination *asks whether there exists two parameters $k > 0$, $l > 0$ such that the TLG $\bar{v}(k, l) = [k \cdot a_1, \ldots, k \cdot a_n, l \cdot b_2, \ldots, l \cdot b_m; k \cdot T_u + l \cdot (T_v - b_1)]$ with fanins $(x_1, \ldots, x_n, y_2, \ldots, y_m)$ satisfies $f_{\bar{v}}(x_1, \ldots, x_n, y_2, \ldots, y_m) = f_v(f_u(x_1, \ldots, x_n), y_2, \ldots, y_m)$ for all assignments to variables $(x_1, \ldots, x_n, y_2, \ldots, y_m)$.*

Figure 1 illustrates the above second formulation. Note that by collapsing u to v via their linear combination we can effectively reduce the dimensions of parameter space from $n + m$ to 2. Below we discuss the feasibility conditions of the special collapsing under linear combination. We first assume $b_1 > 0$ and $\{x_1, \ldots, x_n\} \cap \{y_2, \ldots, y_m\} = \emptyset$ (i.e., fanin variables of u and v are disjoint). In Section 3.3, the cases of $b_1 < 0$ and $\{x_1, \ldots, x_n\} \cap \{y_2, \ldots, y_m\} \neq \emptyset$ (i.e., u and v share common fanin variables) will be discussed.

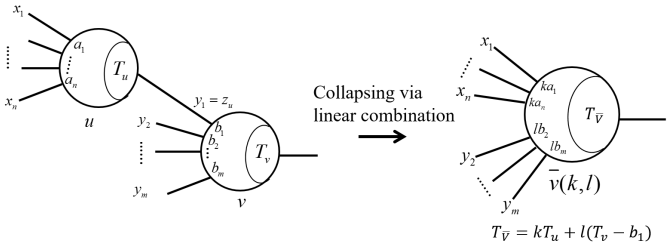

Figure 1: Collapse two TLGs via linear combination.

Example 1. *For a concrete example of the problem formulation, consider TLGs $u = [4, 3; 5]$ with fanins (x_1, x_2) and $v = [2, 1; 3]$ with fanins (y_1, y_2), where $y_1 = z_u$. We ask whether there exist feasible parameters (k, l) to collapse u to v via their linear combination $\bar{v}(k, l) = [4k, 3k, l; 5k+l]$. The answer to this question is to be obtained in Section 3.2.*

3.2 Collapsing Feasibility

To facilitate discussion, we define three functions

$$f_u^+(x_1, \ldots, x_n) \stackrel{\text{def}}{=} \sum_{i=1}^{n} a_i x_i, \qquad f_v^-(y_2, \ldots, y_m) \stackrel{\text{def}}{=} \sum_{j=2}^{m} b_j y_j,$$

$$f_{v \circ u}(x_1, \ldots, x_n, y_2, \ldots, y_m) \stackrel{\text{def}}{=} f_v(f_u(x_1, \ldots, x_n), y_2, \ldots, y_m),$$

and four sets of Boolean assignments

$$\phi_1 \stackrel{\text{def}}{=} \{(\beta_2, \ldots, \beta_m) \mid f_v^-(\beta_2, \ldots, \beta_m) \leq T_v - b_1 - 1\},$$

$$\phi_2 \stackrel{\text{def}}{=} \{(\beta_2, \ldots, \beta_m) \mid f_v^-(\beta_2, \ldots, \beta_m) \geq T_v\},$$

$$\phi_3 \stackrel{\text{def}}{=} \{(\beta_2, \ldots, \beta_m) \mid T_v - b_1 \leq f_v^-(\beta_2, \ldots, \beta_m) \leq T_v - 1\},$$

$$\phi_4 \stackrel{\text{def}}{=} \{(\alpha_1, \ldots, \alpha_n) \mid f_u^+(\alpha_1, \ldots, \alpha_n) \leq T_u - 1\},$$

for $(\alpha_1, \ldots, \alpha_n) \in \mathbb{B}^n$ and $(\beta_2, \ldots, \beta_m) \in \mathbb{B}^{m-1}$.

Observe that for any assignment in ϕ_1 makes $f_{v \circ u} = 0$, any assignment in ϕ_2 makes $f_{v \circ u} = 1$, and any assignment in ϕ_3 makes the valuation of $f_{v \circ u}$ purely determined by f_u. If $\phi_3 = \emptyset$, then $f_{v \circ u} = f_v$ for all assignments to variables $(x_1, \ldots, x_n, y_2, \ldots, y_m)$, i.e., the valuation of z_v does not depend on that of z_u. In the following derivation, this futile situation is excluded.

The necessary and sufficient conditions for collapsing u to v by a linear combination of weights are stated in Theorem 1.

THEOREM 1. *Given two TLGs* $u = [a_1, \ldots, a_n; T_u]$ *with fanins* $(x_1, \ldots, x_n)$ *and* $v = [b_1, \ldots, b_m; T_v]$, $b_1 > 0$, *with fanins* $(y_1, \ldots, y_m)$, $y_1 = z_u$, *and* $\{x_1, \ldots, x_n\} \cap \{y_2, \ldots, y_m\} = \emptyset$, *collapsing* u *to* v *via linear combination under parameter* (k, l) *yielding* $\bar{v} = [ka_1, \ldots, ka_n, lb_2, \ldots, lb_m; kT_u + l(T_v - b_1)]$ *if and only if parameters* (k, l) *satisfy the following inequalities whose side conditions are met.*

$$l(T_v - b_1 - \max_{\beta \in \phi_1}\{f_v^-\}) \geq k(\max\{f_u^+\} - T_u) + 1, \qquad \text{if } \phi_1 \neq \emptyset$$

$$l(\min_{\beta \in \phi_2}\{f_v^-\} + b_1 - T_v) \geq k(T_u - \min\{f_u^+\}), \qquad \text{if } \phi_2 \neq \emptyset$$

$$k(T_u - \max_{\alpha \in \phi_4}\{f_u^+\}) \geq l(\max_{\beta \in \phi_3}\{f_v^-\} + b_1 - T_v) + 1,$$

We remark that it is difficult to compute the coefficients in the inequalities of Theorem 1. For example, to compute $\max_{\beta \in \phi_1}\{f_v^-\}$, one needs to solve the *subset sum problem*, which is NP-complete [4]. This difficulty may be addressed by considering Theorem 2, which states a set of sufficient conditions and is much more efficient to compute.

THEOREM 2. *Given two TLGs* $u = [a_1, \ldots, a_n; T_u]$ *with fanins* $(x_1, \ldots, x_n)$ *and* $v = [b_1, \ldots, b_m; T_v]$, $b_1 > 0$, *with fanins* $(y_1, \ldots, y_m)$, $y_1 = z_u$, *and* $\{x_1, \ldots, x_n\} \cap \{y_2, \ldots, y_m\} = \emptyset$, *collapsing* u *to* v *via linear combination under parameter* (k, l) *yielding* $\bar{v} = [ka_1, \ldots, ka_n, lb_2, \ldots, lb_m; kT_u + l(T_v - b_1)]$ *if parameters* (k, l) *satisfy the following inequalities whose side conditions are met.*

$$l \geq k(\max\{f_u^+\} - T_u) + 1, \qquad \text{if } \phi_1 \neq \emptyset$$

$$lb_1 \geq k(T_u - \min\{f_u^+\}), \qquad \text{if } \phi_2 \neq \emptyset$$

$$k \geq l(b_1 - 1) + 1,$$

Notice that, unlike those in Theorem 1, the max and min operations in Theorem 2 range over all input assignments and do not need to be constrained by ϕ_1 and ϕ_2. Hence, the coefficients, $\max\{f_u^+\}$ and $\min\{f_u^+\}$, in these inequalities of Theorem 2 can be computed in time linear to the fanin size of the TLG, which enhances computational efficiency of collapse operation. A trade-off between the identification power of threshold functions and computational complexity is provided by Theorems 1 and 2.

PROOF. To prove the above two theorems, we search for the conditions such that $f_{v \circ u}$ equals $f_{\bar{v}}$ under all assignments $(\alpha_1, \ldots, \alpha_n, \beta_2, \ldots, \beta_m)$ for variables $(x_1, \ldots, x_n, y_2, \ldots, y_m)$. There are three cases to discuss: First, $(\beta_2, \ldots, \beta_m) \in \phi_1$, i.e., $f_v^-(\beta_2, \ldots, \beta_m) \leq T_v - b_1 - 1$. Second, $(\beta_2, \ldots, \beta_m) \in \phi_2$, i.e., $f_v^-(\beta_2, \ldots, \beta_m) \geq T_v$. Third, $(\beta_2, \ldots, \beta_m) \in \phi_3$, i.e., $T_v - b_1 \leq f_v^-(\beta_2, \ldots, \beta_m) \leq T_v - 1$. Their corresponding derivations are obtained in Lemmas 1 to 3. ■

LEMMA 1. *For every assignment* $(\alpha_1, \ldots, \alpha_n, \beta_2, \ldots, \beta_m)$ *to* $(x_1, \ldots, x_n, y_2, \ldots, y_m)$ *with* $(\beta_2, \ldots, \beta_m) \in \phi_1$, *i.e.,* $f_v^-(\beta_2, \ldots, \beta_m) \leq T_v - b_1 - 1$,

iff-condition : $f_{v \circ u} = f_{\bar{v}} \iff l(T_v - b_1 - \max_{\beta \in \phi_1}\{f_v^-\}) \geq k(\max\{f_u^+\} - T_u) + 1$

if-condition : $f_{v \circ u} = f_{\bar{v}} \Longleftarrow l \geq k(\max\{f_u^+\} - T_u) + 1$

PROOF. For any assignment in ϕ_1, we have $f_{v \circ u} = 0$. In the following derivation, $f_u^+ > T_u$ is assumed, since if $f_u^+ \leq T_u$, $f_{\bar{v}} = f_{v \circ u}$ trivially.

$$\begin{aligned}
f_{\bar{v}} = 0 &\iff kf_u^+ + lf_v^- < kT_u + l(T_v - b_1) \\
&\iff k(f_u^+ - T_u) < l(T_v - b_1 - f_v^-) \\
&\iff \frac{k}{l} < \frac{T_v - b_1 - f_v^-}{f_u^+ - T_u} \\
&\iff \frac{k}{l} < \min\{\frac{T_v - b_1 - f_v^-}{f_u^+ - T_u}\} \\
&\iff \frac{k}{l} < \frac{T_v - b_1 - \max_{\beta \in \phi_1}\{f_v^-\}}{\max\{f_u^+\} - T_u} \\
&\Longleftarrow \frac{k}{l} < \frac{1}{\max\{f_u^+\} - T_u}
\end{aligned}$$

From the above derivation, we have

$$f_{v \circ u} = f_{\bar{v}} \iff l(T_v - b_1 - \max_{\beta \in \phi_1}\{f_v^-\}) \geq k(\max\{f_u^+\} - T_u) + 1$$

$$f_{v \circ u} = f_{\bar{v}} \Longleftarrow l \geq k(\max\{f_u^+\} - T_u) + 1$$

■

EXAMPLE 2. *Continue Example 1. By* $\phi_1 \neq \emptyset$, *the conditions in Lemma 1 should be satisfied by parameter* (k, l). *The conditions are*

$$f_{v \circ u} = f_{\bar{v}} \iff l(3 - 2 - 0) \geq k(7 - 5) + 1$$
$$f_{v \circ u} = f_{\bar{v}} \Longleftarrow l \geq k(7 - 5) + 1$$

Suppose the condition $l \geq 2k + 1$ *is violated, e.g., by setting* $l = 2k$. *The resulting TLG* $\bar{v}$ *becomes* $[4, 3, 2; 7]$. *As predicted by Lemma 1, a discrepancy must occur. Indeed, we can find* $(x_1, x_2, y_2) = (1, 1, 0)$ *such that* $f_{\bar{v}} = 1$ *and* $f_{v \circ u} = 0$.

LEMMA 2. *For every assignment* $(\alpha_1, \ldots, \alpha_n, \beta_2, \ldots, \beta_m)$ *to* $(x_1, \ldots, x_n, y_2, \ldots, y_m)$ *with* $(\beta_2, \ldots, \beta_m) \in \phi_2$, *i.e.,* $f_v^-(\beta_2, \ldots, \beta_m) \geq T_v$:

iff-condition : $f_{v \circ u} = f_{\bar{v}} \iff l(\min_{\beta \in \phi_2}\{f_v^-\} + b_1 - T_v) \geq k(T_u - \min\{f_u^+\})$

if-condition : $f_{v \circ u} = f_{\bar{v}} \Longleftarrow lb_1 \geq k(T_u - \min\{f_u^+\})$

PROOF. For any assignment in ϕ_2, we have $f_{v \circ u} = 1$. In the following derivation, $f_u^+ < T_u$ is assumed, since if $f_u^+ \geq T_u$, $f_{\bar{v}} = f_{v \circ u}$ trivially.

$$\begin{aligned}
f_{\bar{v}} = 1 &\iff kf_u^+ + lf_v^- \geq kT_u + l(T_u - b_1) \\
&\iff k(f_u^+ - T_u) \geq l(T_v - b_1 - f_v^-) \\
&\iff \frac{k}{l} \leq \frac{f_v^- + b_1 - T_v}{T_u - f_u^+} \\
&\iff \frac{k}{l} \leq \min\{\frac{f_v^- + b_1 - T_v}{T_u - f_u^+}\} \\
&\iff \frac{k}{l} \leq \frac{\min_{\beta \in \phi_2}\{f_v^-\} + b_1 - T_v}{T_u - \min\{f_u^+\}} \\
&\Longleftarrow \frac{k}{l} \leq \frac{b_1}{T_u - \min\{f_u^+\}}
\end{aligned}$$

From the above derivation, we have

$$f_{v \circ u} = f_{\bar{v}} \iff l(\min_{\beta \in \phi_2}\{f_v^-\} + b_1 - T_v) \geq k(T_u - \min\{f_u^+\})$$

$$f_{v \circ u} = f_{\bar{v}} \Longleftarrow lb_1 \geq k(T_u - \min\{f_u^+\})$$

■

EXAMPLE 3. *Continue Example 2. Since $\phi_2 = \emptyset$, the conditions in Lemma 2 need not be imposed for parameter (k, l).*

LEMMA 3. *For every assignment $(\alpha_1, \ldots, \alpha_n, \beta_2, \ldots, \beta_m)$ to $(x_1, \ldots, x_n, y_2, \ldots, y_m)$ with $(\beta_2, \ldots, \beta_m) \in \phi_3$, i.e., $T_v - b_1 \leq f_v^-(\beta_2, \ldots, \beta_m) \leq T_v - 1$:*

iff-condition *:* $f_{v \circ u} = f_{\bar{v}} \iff$
$$k(T_u - \max_{\alpha \in \phi_4}\{f_u^+\}) \geq l(\max_{\beta \in \phi_3}\{f_v^-\} + b_1 - T_v) + 1$$

if-condition *:* $f_{v \circ u} = f_{\bar{v}} \Longleftarrow k \geq l(b_1 - 1) + 1$

PROOF. For any assignment in ϕ_3, we have $kf_u^+ + lf_v^- \geq kT_u + l(T_v - b_1) \iff f_u^+ \geq T_u$. In the following derivation, $f_u^+ < T_u$ is assumed, since if $f_u^+ \geq T_u$, the assumption $f_v^- \geq T_v - b_1$ implies $kf_u^+ + lf_v^- \geq kT_u + l(T_v - b_1)$, and $f_{\bar{v}} = f_{v \circ u}$. Under the assumption of $f_u^+ < T_u$, $f_{v \circ u} = 0$.

$$\begin{aligned}
f_{\bar{v}} = 0 \quad &\iff \quad kf_u^+ + lf_v^- < kT_u + l(T_v - b_1) \\
&\iff \quad k(f_u^+ - T_u) < l(T_v - b_1 - f_v^-) \\
&\iff \quad \frac{k}{l} > \frac{f_v^- + b_1 - T_v}{T_u - f_u^+} \\
&\iff \quad \frac{k}{l} > \max\{\frac{f_v^- + b_1 - T_v}{T_u - f_u^+}\} \\
&\iff \quad \frac{k}{l} > \frac{\max_{\beta \in \phi_3}\{f_v^-\} + b_1 - T_v}{T_u - \max_{\alpha \in \phi_4}\{f_u^+\}} \\
&\Longleftarrow \quad \frac{k}{l} > b_1 - 1
\end{aligned}$$

From the above derivation, we have

$$f_{v \circ u} = f_{\bar{v}} \iff k(T_u - \max_{\alpha \in \phi_4}\{f_u^+\}) \geq l(\max_{\beta \in \phi_3}\{f_v^-\} + b_1 - T_v) + 1$$
$$f_{v \circ u} = f_{\bar{v}} \Longrightarrow k \geq l(b_1 - 1) + 1$$

■

EXAMPLE 4. *Continue Example 3. Because $\phi_3 \neq \emptyset$, the conditions in Lemma 3 should be satisfied by parameter (k, l). The conditions are*

$$f_{v \circ u} = f_{\bar{v}} \iff k(5 - 4) \geq l(1 + 2 - 3) + 1$$
$$f_{v \circ u} = f_{\bar{v}} \Longleftarrow k \geq l(2 - 1) + 1$$

The condition $k \geq 1$ should not be violated since we consider $k > 0, l > 0$ in Problem Formulation 2. From the derived iff-constraints (i.e., one from Lemma 1 and the other from Lemma 3) in Examples 2 to 4, the two iff-conditions can be satisfied, e.g., by $(k, l) = (1, 3)$, which asserts that u can be collapsed to v yielding $\bar{v} = [4, 3, 3; 8]$. On the other hand, the derived if-constraints (one from Lemma 1 and the other from Lemma 3) in Examples 2 to 4 together yield no solution due to their under-approximation of (k, l) solutions.

We remark that the merging operations proposed in [3] are special cases of the collapse operation formulated above. For example, the AND *gate-based merging* in [3], which considers to "merge" an AND gate v with one of its fanins u, is equivalent to collapsing u to v in our formulation. Theorem 1 and 2 can be applied to derive feasible parameters to collapse u to v via their linear combination.

Due to the linearity of the inequality constraints, an advantage of the proposed collapsing method lies in that, after coefficients in the inequalities are computed, analytic solutions for (k, l) can be obtained. We remark that, in general, there may be multiple feasible solutions for (k, l), and those with smaller magnitude for k and l are preferred since the collapsed TLG would have smaller weights and threshold values. A possible approach to derive feasible solutions analytically is to find the intersecting point of these constraints and rounding it up to integers. For example, suppose two inequalities $l \geq k(\max\{f_u^+\} - T_u) + 1$ and $k \geq l(b_1 - 1) + 1$ are derived based on the if-conditions, one can solve for the intersecting point of the two lines analytically. Let the coordinates of the intersecting point be (l^*, k^*). A feasible (k, l) combination can be derived by first rounding l^* up to $\lceil l^* \rceil$ and substituting $\lceil l^* \rceil$ into $k = l(b_1 - 1) + 1$ to obtain the corresponding k coordinate. Such analytic search results in a fast collapsing computation. Also, observe that in Examples 2 to 4, if u and v are canonicalized to $[1, 1; 2]$ before collapsing, both the iff-conditions and the if-conditions can yield the solution $(k, l) = (1, 1)$ to collapse u to v resulting in $\bar{v} = [1, 1, 1; 3]$. Applying canonicalization in [11] may enhance the feasibility of collapse operation.

3.3 Extension to Negative Weight and Non-disjoint Fanin

Theorems 1 and 2 assume $b_1 > 0$ and $\{x_1, \ldots, x_n\} \cap \{y_2, \ldots,$
$y_m\} = \emptyset$. We show how to handle the cases when $b_1 < 0$ and $\{x_1, \ldots, x_n\} \cap \{y_2, \ldots, y_m\} \neq \emptyset$ by an example. The basic operations to complement a fanin variable and invert a TLG can be found in [14].

EXAMPLE 5. *Let $u = [1, 1; 2]$ with fanins (x_1, x_2) and $v = [-1, -1, 0]$ with fanins (y_1, y_2), where $y_1 = z_u$ and $y_2 = x_2$. We first make the negative weight positive by complementing y_1, resulting in $v^* = [1, -1; 1]$. The inverter generated by complementing y_1 is then combined with u, yielding $u^* = [-1, -1; -1]$. To collapse u to v, it is equivalent to collapse u^* to v^*, where the weight of y_1 is inverted to 1, and Theorem 1 may be applied by first assuming that x_2 and y_2 are different variables. In this example, one can verify that $(k, l) = (1, 2)$ is a feasible solution. As a result, the collapsed TLG is $\bar{v} = [-1, -1, -2; -1]$ with fanins (x_1, x_2, y_2). Since x_2 and y_2 are actually the same variable, their weights should be summed up and the resulting TLG is $\bar{v} = [-1, -3; -1]$ with fanins (x_1, x_2).*

4. TRANSLATING TL CIRCUIT TO PSEUDO BOOLEAN CONSTRAINTS

As an analogy to Tseitin's conversion translating Boolean logic circuits to conjunctive normal form (CNF) formulas [21], we propose a linear-time translation from TL circuits to pseudo Boolean (PB) constraints. Our translation converts every gate in a TL circuit to exactly two PB constraints, both of which are of length linear to the fanin size of the TLG. The conjunction of all the PB constraints gives the consistency condition of variable assignments on the entire circuit. Our translation may facilitate formal reasoning on TL circuits with PB constraint solving, similar to reasoning on Boolean logic circuits with satisfiability (SAT) solving. The conversion is detailed as follows.

Given a TLG $u = [a_1, \ldots, a_n; T]$ with input variables $(x_1,$ $\ldots, x_n)$ and output variable y, we would like to derive PB constraints stating the relation $y \leftrightarrow (\sum_{i=1}^n a_i x_i \geq T)$, which is unfortunately not in the right format of PB constraints. The following theorem provides a solution.

THEOREM 3. *Given a TLG $u = [a_1, \ldots, a_n; T]$ with input variables $(x_1, \ldots, x_n)$ and output variable y, a truth assignment to variables $x_1, \ldots, x_n, y$ satisfies $y \leftrightarrow \sum_{i=1}^{n} a_i x_i \geq T$ if and only if it satisfies the following two pseudo Boolean constraints:*

$$(M - T - 1)y + (T - 1 - \sum_{i=1}^{n} a_i x_i) \geq 0 \quad (1)$$

$$(\sum_{i=1}^{n} a_i x_i - T) + (T - m)(1 - y) \geq 0, \quad (2)$$

where M (resp. m) is the sum of all positive (resp. negative) weights among $\{a_1, \ldots, a_n\}$. M (resp. m) is defined to be 0 if all weights are non-positive (resp. non-negative).

PROOF. Let $A \subseteq \mathbb{B}^{n+1}$ be the set of valuations to variables $(x_1, \ldots, x_n, y)$ that satisfy $y \leftrightarrow \sum_{i=1}^{n} a_i x_i \geq T$; let $B \subseteq \mathbb{B}^{n+1}$ be the set of valuations to variables $(x_1, \ldots, x_n, y)$ that satisfy both Eq. (1) and (2). We show $A = B$ by establishing $A \subseteq B$ and $B \subseteq A$ as follows.

To show $A \subseteq B$, consider an arbitrary $\tilde{z} = (\alpha_1, \ldots, \alpha_n, \beta) \in A$. There are two cases: When $\sum_{i=1}^{n} a_i \alpha_i \geq T$ and $\beta = 1$, Eq. (1) is satisfied by $\tilde{z}$ because by substituting $\tilde{z}$ into Eq. (1) we have $M - \sum_{i=1}^{n} a_i \alpha_i \geq 0$, which holds since M is defined to be the summation of all positive a_i. Also Eq. (2) is satisfied by $\tilde{z}$ because by substituting $\tilde{z}$ into Eq. (2) we have $\sum_{i=1}^{n} a_i \alpha_i \geq T$, which is our assumption. When $\sum_{i=1}^{n} a_i \alpha_i \leq T - 1$ and $\beta = 0$, Eq. (1) is satisfied by $\tilde{z}$ because by substituting $\tilde{z}$ into Eq. (1) we have $\sum_{i=1}^{n} a_i \alpha_i \leq T - 1$, which is our assumption. Eq. (2) is satisfied by $\tilde{z}$ because by substituting $\tilde{z}$ into Eq. (2) we have $\sum_{i=1}^{n} a_i \alpha_i - m \geq 0$, which holds since m is defined to be the summation of all negative a_i. From the above two cases, we have shown that $\tilde{z} \in B$, and thus $A \subseteq B$.

To show $B \subseteq A$, consider an arbitrary $\tilde{z} = (\alpha_1, \ldots, \alpha_n, \beta) \in B$. There are two cases: When $\sum_{i=1}^{n} a_i \alpha_i \geq T$, Eq. (1) implies $y = 1$ because $T - 1 - \sum_{i=1}^{n} a_i x_i < 0$, and under $y = 1$ Eq. (2) is automatically satisfied. Hence, $\tilde{z} \in A$. When $\sum_{i=1}^{n} a_i x_i \leq T - 1$, Eq. (2) implies $y = 0$ because $\sum_{i=1}^{n} a_i x_i - T < 0$, and under $y = 0$ Eq. (1) is satisfied automatically. Hence, $\tilde{z} \in A$. From the above two cases, we have shown that $\tilde{z} \in A$, and thus $B \subseteq A$. ■

Prior work [7] translates a TLG into its maximally factored form and then converts it into Boolean expression diagrams (BED) [9]. The converted BED is then transformed to BDD for further Boolean reasoning. However, BDD representation tends to suffer from the memory explosion issue when the number of primary inputs is large. The method may not be scalable. Prior work [23] applies a path search approach to enumerate the product terms for a sum-of-product (SOP) expression of a TLG. The SOP expression is then converted to a CNF formula for SAT solving based Boolean reasoning. It also has the memory explosion issue, especially when the fanin size of a TLG is large, the SOP expression can blow up. In contrast, our method can translate a TLG in time linear to the fanin size, resulting in a more efficient translation and compact representation of TLG.

5. APPLICATIONS

5.1 Synthesis

Given a TL circuit $G = (V, E)$, we apply the collapse operation to minimize the number of vertices. A TLG u can be eliminated from the circuit if it can be collapsed into all of its fanouts. We remark that minimizing the number of gates of a TL circuit may not be the only optimization objective. There can be other cost metrics in terms of, e.g., fanin sizes, weight values, and threshold values. If large fanin sizes, and large weight and threshold values are a concern, additional constraints can be imposed to restrict the collapse operation.

Figure 2 sketches the procedure *CollapseNtk*. When a TLG u can be collapsed to all of its fanouts, the procedure collapses them by calling *CollapseNode* in Figure 3, where parameters (k, l) are computed by *CalKL*, and a new TLG is created by *CreateNode*. The collapsibility check in line06 involves calling *CalKL* several times, one for each fanout of u. In our implementation, we resort to the sufficient conditions stated in Theorem 2 due to its low computation cost. Our empirical experience suggests that the iff-conditions in Theorem 1 offer negligible improvement in gate count reduction, and thus the if-conditions already provide good approximation to the sufficient and necessary criteria for collapse. A TLG is marked if none of its fanin can be collapsed to it; *CollapseNtk* terminates when all of the TLGs are marked.

CollapseNtk has an additional parameter B that constrains the fanout size of a gate when it is considered to be collapsed into its fanouts, in line05 of Figure 2. We apply a strategy of iteratively incrementing the bound in *CollapseIter*. As to be seen in the experiments, compared to collapsing without limitation on the fanout size, i.e., running *CollapseNtk* with $B = \infty$ directly, *CollpseIter* has better performance in terms of minimizing gate count. This phenomenon can be explained by the fact that if no limitation is imposed from the beginning, a gate with large fanout size would be collapsed to all its fanouts earlier if collapsible, resulting in a large number of collapsed gates, which are more difficult to be collapsed with other gates. Instead, if we set a bound to the fanout size and iteratively increases the bound, we can mitigate the increasing non-collapsibility.

```
CollapseNtk
  input: threshold logic network G = (V, E) and a bound B
  output: collapsed network G' = (V', E')
  begin
  01   unmark every v ∈ V;
  02   while (not all v ∈ V are marked)
  03     foreach v ∈ V
  04       foreach fanin u of v
  05         if |fanouts(u)| ≤ B
  06           if u can be collapsed to all of its fanouts
  07             foreach fanout t of u
  08               w := CollapseNode(u,t);
  09               unmark w;
  10               V := V\{t} ∪ {w};
  11             V := V\{u};
  12             continue; //to next v
  13         if every fanin of v cannot be collapsed to v
  14           mark v;
  15   return (V, E);
  end
```

Figure 2: Algorithm: Collapse TL circuit.

5.2 Verification

Given two TL circuits, we may apply our TLG-to-PB-constraints conversion for their equivalence checking. A miter can be built to assert the equivalence relation between two circuits by disjuncting the XORs of corresponding out-

CollapseNode
input: two TLGs u and v
output: collapsed TLG w if collapsible, or NULL otherwise
begin
01 $(k, l) := CalKL(u,v)$;
02 **if** $k > 0$ and $l > 0$
03 $w := CreateNode(u,v,k,l)$;
04 **return** w;
05 **else**
06 **return NULL**;
end

Figure 3: Algorithm: Collapse two TLGs.

CollapseIter
input: threshold logic network $G = (V, E)$ and a bound B
output: iteratively collapsed network $G' = (V', E')$
begin
01 $i := 1$;
02 **while** $i \leq B$;
03 $(V,E) := CollapseNtk(G(V, E),i)$;
04 $i := i + 1$;
05 **return** (V,E);
end

Figure 4: Algorithm: Iterative collapsing.

put pairs. For each TLG in the miter, Theorem 3 is applied to translate it into PB constraints. For the added output equivalence circuitry, it can be first translated by Tseitin conversion into a set of clauses in CNF and then further translated for each clause to its PB constraint. Having all the PB constraints are obtained, we set the objective function to maximize the value of the miter output variable. The maximum value is 0 if and only if the two TL circuits under verification are equivalent.

6. EXPERIMENTAL RESULTS

The collapsing-based synthesis and equivalence verification algorithms discussed in Section 5 were implemented in the `ABC` environment [1]. The experiments were conducted on a Linux machine with a Xeon 2.3 GHz CPU and 200 GB RAM. `ISCAS` and `ITC` benchmark suits were selected for experiments. In our experiments, we refer to the architecture parameters of TrueNorth chip [13] and impose a fanin size limitation of 256 and a weight value limitation in range $[-255, 255]$ for a TLG.

The proposed collapse operation can be directly applied to AIGs as an AIG node can be easily translated to a TLG. For example, an AND gate is translated to $[1, 1; 2]$ and a NAND gate is translated to $[-1, -1; -1]$. We experimented on collapsing a functionally reduced AIG (synthesized by command `fraig` in `ABC`) into a TL circuit, and the results are shown in Table 1. Columns 4 and 5 report the numbers of AND gates and logic levels of the AIG circuits, respectively; Columns 6, 7, and 8 report the number of TLGs, the number of logic levels of the TL circuits, and the CPU time, respectively, for direct collapsing without fanout bounds (as discussed in Section 5.1); Columns 9, 10, and 11 report the number of TLGs, the number of logic levels of the TL circuits, and the runtime, respectively, for iterative collapsing with incrementing fanout bounds. The numbers in parentheses listed in Columns 6 and 9 (resp. Columns 7 and 10) are the ratios of gate counts (resp. level counts) of the collapsed TL circuits to those of original AIG circuits. Without the increment strategy of fanout bounds, the collapse operation achieves an average of 50% reduction in gate count and 28% reduction in logic level; with the increment strategy, the reduction ratios are further enhanced to 55% in gate count and 30% in logic level.

To compare the synthesis quality with the methods proposed in [3], we also apply our iterative collapsing technique to five largest benchmarks experimented in [3]. The results are shown in Table 2. Columns 2 and 3, with data repeated from [3], report the numbers of AND gates in the benchmark circuits and the numbers of TLGs in the synthesized circuits. Columns 4 and 5 report the numbers of AND gates in the benchmark circuits (synthesized by `resyn2` script in `ABC`, as described in [3]) and the numbers of TLGs in the collapsed TL circuits. Apart from the slight mismatch between the numbers in Columns 2 and 4, an average of 43% reduction in gate count was achieved in [3] while an average of 56% reduction was achieved by our collapsing-based synthesis technique, confirming the generality and effectiveness of our formulation.

We further experiment on collapsing the TL circuits synthesized by [15] to evaluate the room for collapsing on optimized TL circuits. The overall experimental flow is as follows.First, threshold logic synthesis method (`ABC &if` mapper) used in [15] is applied to transform benchmark circuits into TL circuits. Second, the algorithm *CollapseIter* is applied to the synthesized TL circuits to collapse TLGs. Two collapse strategies were applied: one with fanout bound B set to infinity and the other with B incrementing from 1 up to 100. After collapsing, equivalence checking based on the proposed TLG-to-PB conversion is applied to verify the equivalence between TL circuits before and after collapsing. A miter is built to assert the equivalence between the collapsed network and the original one. Moreover, we reimplemented the state-of-the-art method, *path search with weight ordering* [23], called `weight order` in our discussion, to compare and evaluate our PB translation method. The miter is translated into a set of clauses by `weight order` and into a set of PB constraints by our translation. SAT solver `Minisat` and PB solver `Minisat+` were adopted to solve the CNF and PB constraints, respectively. In addition to the two methods, we also applied command `cec`, a state-of-the-art combinational equivalent checking (CEC) tool, in `ABC` for equivalence checking the AIGs translated by `weight order` from TLGs to compare.

Table 3 shows the results of collapse operation on TL circuits optimized by [15]. Columns 4, 5, and 6 report the number of TLGs, the number of levels of the TL circuits synthesized by [15], and the CPU time, respectively; Columns 7, 8, and 9 report the number of TLGs, the number of logic levels of the TL circuits, and the CPU time, respectively, for direct collapsing without fanout bounds; Columns 10, 11, and 12 report the number of TLGs, the number of logic levels of the TL circuits, and the CPU time, respectively, for iterative collapsing with incrementing fanout bounds.

The numbers in parentheses listed in Columns 7 and 10 (resp. Columns 8 and 11) are the ratios of gate counts (resp. level counts) of the collapsed TL circuits to those of TL circuits optimized by [15]. On top of the synthesized circuits by [15], the collapse operation obtains an average of 16% reduction in the number of TLGs with runtime less than

Table 1: Statistics of collapsing AIG circuits

benchmarks profile			statistics of AIG		statistics of collapsed TLC (w/o ite)			statistics of collapsed TLC (ite)		
circuit	#pi	#po	#AND	#level	#TLG	#level	time (s)	#TLG	#level	time (s)
c3540	50	22	1028 (1.00)	40 (1.00)	514 (0.50)	33 (0.83)	0.00	418 (0.41)	29 (0.73)	0.03
c5315	178	123	1741 (1.00)	38 (1.00)	736 (0.42)	27 (0.71)	0.00	684 (0.39)	23 (0.61)	0.05
c6288	32	32	2334 (1.00)	120 (1.00)	1407 (0.60)	93 (0.78)	0.01	1404 (0.60)	95 (0.79)	0.17
c7552	207	108	1961 (1.00)	29 (1.00)	991 (0.51)	22 (0.76)	0.01	846 (0.43)	19 (0.66)	0.08
s5378	35	49	1362 (1.00)	19 (1.00)	578 (0.42)	14 (0.74)	0.00	537 (0.39)	11 (0.58)	0.03
s9234.1	36	39	1804 (1.00)	33 (1.00)	835 (0.46)	18 (0.55)	0.01	762 (0.42)	18 (0.55)	0.06
s13207	31	121	2605 (1.00)	33 (1.00)	1213 (0.47)	21 (0.64)	0.01	1190 (0.46)	20 (0.61)	0.08
s15850	14	87	3330 (1.00)	46 (1.00)	1601 (0.48)	34 (0.74)	0.01	1479 (0.44)	32 (0.70)	0.10
s35932	35	320	10124 (1.00)	14 (1.00)	5078 (0.50)	9 (0.64)	0.03	4758 (0.47)	8 (0.57)	0.19
s38417	28	106	9062 (1.00)	30 (1.00)	4747 (0.52)	21 (0.70)	0.03	4388 (0.48)	19 (0.63)	0.32
s38584	12	278	11646 (1.00)	34 (1.00)	4981 (0.43)	22 (0.65)	0.04	4639 (0.40)	20 (0.59)	0.32
b04	11	8	534 (1.00)	23 (1.00)	300 (0.56)	17 (0.74)	0.00	284 (0.53)	17 (0.74)	0.02
b12	5	6	996 (1.00)	17 (1.00)	498 (0.50)	12 (0.71)	0.00	438 (0.44)	15 (0.88)	0.03
b14	32	54	5609 (1.00)	65 (1.00)	2866 (0.51)	49 (0.75)	0.02	2565 (0.46)	53 (0.82)	0.26
b15	36	70	8158 (1.00)	65 (1.00)	4028 (0.49)	47 (0.72)	0.05	3667 (0.45)	44 (0.68)	0.29
b17	37	97	26389 (1.00)	93 (1.00)	13379 (0.51)	66 (0.71)	0.17	12027 (0.46)	73 (0.78)	1.25
b18	37	23	77757 (1.00)	132 (1.00)	39733 (0.51)	98 (0.74)	0.42	35343 (0.45)	87 (0.66)	4.70
b19	24	27	156224 (1.00)	136 (1.00)	80621 (0.52)	105 (0.77)	0.87	70765 (0.45)	85 (0.63)	10.18
b20	32	22	11552 (1.00)	66 (1.00)	5929 (0.51)	47 (0.71)	0.05	5284 (0.46)	58 (0.88)	0.51
b21	32	22	11728 (1.00)	70 (1.00)	6017 (0.51)	54 (0.77)	0.06	5381 (0.46)	59 (0.84)	0.53
b22	32	22	17614 (1.00)	68 (1.00)	9071 (0.51)	57 (0.84)	0.09	8028 (0.46)	60 (0.88)	0.94
geomean			(1.00)	(1.00)	(0.50)	(0.72)		(0.45)	(0.70)	

Table 2: Comparison between methods in [3] and iterative collapsing

	statistics in [3]		statistics of collapsed TLC	
circuit	#AND	#TLG	#AND	#TLG
aes_core	20509	10057	19875	7505
wb_conmax	41070	21956	41163	19626
ethernet	57205	35243	43549	19110
des_perf	71327	42719	69500	27316
vga_lcd	88854	55402	90880	50734
geomean	(1.00)	(0.57)	(1.00)	(0.44)

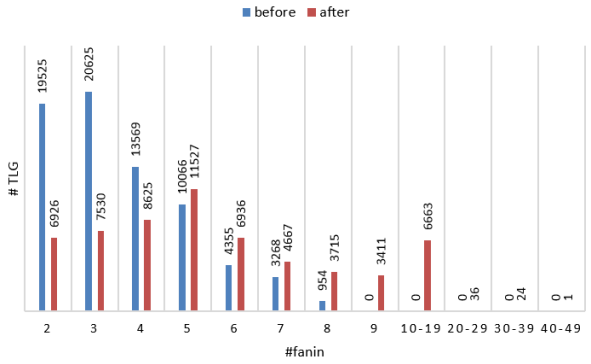

Figure 5: Fanin size distribution of `b19` before vs. after iterative collapsing.

7 seconds for all benchmarks. The results show the effectiveness of the proposed collapse operation based on linear composition. The efficiency lies in fast analytic computation of collapsing conditions. However, the collapse operation achieves no reduction in logic level. This phenomenon could be attributed to the well-optimized `ABC &if` mapper applied in [15]. Since the technology mapper has optimized the circuits in terms of delay, it is hard to further reduce logic levels.

In Figure 5, we use the benchmark circuit `b19` as a representative to show the fanin size distribution of TLGs in TL circuits before and after iterative collapsing. Since the method in [15] can only synthesize TL circuits consisting of TLGs with no greater than 8 fanins, the numbers of TLGs with more than 8 fanins before collapsing are 0. Observe that the numbers of TLGs with less than 5 fanins decrease after collapsing, which is a general phenomenon among all large benchmarks. The proportion of the gates with more than 8 fanins in large benchmarks, such as `b14` to `b22`, can account for over 13% of all gates. This notable proportion reflects the effectiveness of our collapse operation for TLG elimination.

Table 4 (resp. Table 5) shows the results of equivalence verification between circuits before and after collapsing without (resp. with) the iterative fanout bound increment. The three verification techniques mentioned above, i.e., CNF-based, AIG-based (CEC), and PB-based, were evaluated in terms of CPU time and memory usage. Columns 2, 3, and 4 show the runtime in seconds spent in generating files for different techniques; Columns 5, 6, and 7 show the size in KB of the generated files; Columns 8, 9, and 10 show the runtime for equivalence checking. A time (resp. memory) limit of two hours (resp. 1GB) is imposed, and a `TO` (resp. `MO`) in the tables indicates timeout (resp. memory out).

As can be seen from the two tables, TLG-to-PB translation always generates constraints of size linear to the number of TLGs in the TL circuit. This property results in a fast and compact translation. In contrast, the CNF formula size and AIG size resulted from `weight order` translation are sensitive to the SOP representations of TLGs. These translations suffer from memory explosion. On average, the size of CNF formula (resp. AIG) files is 70 (resp. 7) times the size of PB files. In Table 4 the `weight order` translation exceeds the memory limit for benchmarks `b14` to `b22`, whereas in Table 5 it can generate files within memory limit except for `b18` and `b19`. This phenomenon suggests that collapsing with iterative fanout bound increment could synthesize TL

Table 3: Statistics of collapsing TL circuits

benchmarks profile			statistics of TLC synthesized by [15]			statistics of collapsed TLC (w/o ite)			statistics of collapsed TLC (ite)		
circuit	#pi	#po	#TLG	#level	time (s)	#TLG	#level	time (s)	#TLG	#level	time (s)
c3540	50	22	465 (1.00)	13 (1.00)	1.90	397 (0.85)	13 (1.00)	0.00	399 (0.86)	13 (1.00)	0.03
c5315	178	123	732 (1.00)	10 (1.00)	2.58	643 (0.88)	10 (1.00)	0.00	644 (0.88)	10 (1.00)	0.04
c6288	32	32	1424 (1.00)	29 (1.00)	6.04	1105 (0.78)	29 (1.00)	0.00	1102 (0.77)	29 (1.00)	0.11
c7552	207	108	950 (1.00)	10 (1.00)	4.91	716 (0.75)	10 (1.00)	0.00	713 (0.75)	10 (1.00)	0.06
s5378	35	49	578 (1.00)	5 (1.00)	1.64	489 (0.85)	5 (1.00)	0.00	489 (0.85)	5 (1.00)	0.03
s9234.1	36	39	744 (1.00)	7 (1.00)	2.23	635 (0.85)	7 (1.00)	0.00	631 (0.85)	7 (1.00)	0.04
s13207	31	121	1257 (1.00)	8 (1.00)	2.10	1046 (0.83)	8 (1.00)	0.00	1049 (0.83)	8 (1.00)	0.05
s15850	14	87	1630 (1.00)	10 (1.00)	3.24	1360 (0.83)	10 (1.00)	0.00	1356 (0.83)	10 (1.00)	0.08
s35932	35	320	5878 (1.00)	5 (1.00)	1.84	4299 (0.73)	5 (1.00)	0.01	4299 (0.73)	5 (1.00)	0.14
s38417	28	106	4857 (1.00)	8 (1.00)	5.14	4248 (0.87)	8 (1.00)	0.01	4220 (0.87)	8 (1.00)	0.28
s38584	12	278	4391 (1.00)	8 (1.00)	6.24	4000 (0.91)	8 (1.00)	0.01	3999 (0.91)	8 (1.00)	0.19
b04	11	8	270 (1.00)	9 (1.00)	1.67	235 (0.87)	9 (1.00)	0.00	235 (0.87)	9 (1.00)	0.01
b12	5	6	490 (1.00)	5 (1.00)	2.33	445 (0.91)	5 (1.00)	0.00	445 (0.91)	5 (1.00)	0.03
b14	32	54	2680 (1.00)	13 (1.00)	11.51	2253 (0.84)	13 (1.00)	0.01	2218 (0.83)	13 (1.00)	0.21
b15	36	70	4077 (1.00)	16 (1.00)	14.99	3616 (0.89)	16 (1.00)	0.01	3566 (0.87)	16 (1.00)	0.30
b17	37	97	13009 (1.00)	22 (1.00)	37.48	11279 (0.87)	22 (1.00)	0.05	11131 (0.86)	22 (1.00)	0.90
b18	37	23	36370 (1.00)	43 (1.00)	88.31	30732 (0.84)	43 (1.00)	0.14	30180 (0.83)	43 (1.00)	2.90
b19	24	27	72362 (1.00)	46 (1.00)	181.76	61320 (0.85)	46 (1.00)	0.29	60061 (0.83)	46 (1.00)	6.67
b20	32	22	5660 (1.00)	15 (1.00)	18.82	4679 (0.83)	15 (1.00)	0.02	4630 (0.82)	15 (1.00)	0.42
b21	32	22	5660 (1.00)	16 (1.00)	19.29	4730 (0.84)	16 (1.00)	0.02	4628 (0.82)	16 (1.00)	0.44
b22	32	22	8429 (1.00)	16 (1.00)	24.42	7000 (0.83)	16 (1.00)	0.03	6846 (0.81)	16 (1.00)	0.67
geomean			(1.00)	(1.00)		(0.84)	(1.00)		(0.84)	(1.00)	

circuits with simpler TLGs.

The proposed PB-based equivalence checking solved six large benchmarks (`b14-17`, `b20-22`) uniquely in Table 4, demonstrating the unique value of the linear translation. On the other hand, not surprisingly, CEC achieves the best scalability among the three methods due to its powerful exploitation of circuit similarities for verification reduction. A compact translation from TLG to AIG would greatly benefit the usage of CEC and improve the scalability of threshold logic verification, which is left as our future work.

7. CONCLUSIONS AND FUTURE WORK

In this paper, the collapse operation of threshold logic gates has been formulated, and the sufficient and necessary conditions of collapsibility using linear composition have been derived. Moreover, a linear-time translation from a threshold logic gate to PB constraints has been proposed for equivalence verification. Experimental results showed fast and effective TL circuit collapsing as well as memory efficient equivalence checking.Specifically, on top of the optimized TL circuits, an average of 16% gate count reduction is achieved by the collapse operation, and the proposed translation uniquely solves six benchmarks in our experiments. For future work, we would like to study how to combine the collapse operation with other TL operations to form useful scripts for TL circuit optimization, to develop compact translation from TLG to AIG for verification, and to apply TL synthesis for neural network applications.

Acknowledgments

The authors thank Augusto Neutzling for helpful discussions about prior work [15], and Alan Mishchenko and anonymous reviewers for their valuable suggestions. This work was supported in part by the Ministry of Science and Technology of Taiwan under grants 103-2221-E-002-273, 104-2622-8-002-003, and 104-2628-E-002-013-MY3.

8. REFERENCES

[1] Berkeley Logic Synthesis and Verification Group. ABC: A system for sequential synthesis and verification. `http://www.eecs.berkeley.edu/~alanmi/abc/`

[2] V. Beiu, J. Quintana and M. Avedillo. VLSI implementations of threshold logic – A comprehensive survey. *IEEE Trans. on Neural Networks*, 14(5): 1217-1243, 2003.

[3] Y.-C. Chen, R. Wang and Y.-P Chang. Fast synthesis of threshold logic networks with optimization. In *Proc. ASP-DAC*, pp. 486-491, 2016.

[4] T. Cormen, C. Leiserson, R. Rivest and C. Stein. *Introduction to Algorithms*. MIT press, 2009.

[5] L. Gao, F. Alibart and D. Strukov. Programmable CMOS/memristor threshold logic. *IEEE Trans. on Nanotechnology*, 12(2): 115-119, 2013.

[6] T. Gowda, S. Leshner, S. Vrudhula and G. Konjevod. Synthesis of threshold logic circuits using tree matching. In *Proc. ECCTD*, pp. 850-853, 2007.

[7] T. Gowda, S. Vrudhula and G. Konjevod. Combinational equivalence checking for threshold logic circuits. In *Proc. GLSVLSI*, pp. 102-107, 2007.

[8] G.-B. Huang, Q.-Y. Zhu, K.-Z. Mao, C.-K. Siew and P. Saratchandran. Can threshold networks be trained directly? *IEEE Trans. on Circuits and Systems II: Express Briefs*, 53(3): 187-191, 2006.

[9] H. Hulgaard, P. Williams and H. Andersen. Equivalence checking of combinational circuits using Boolean expression diagrams. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 18(7): 903-917, 1999.

[10] A. Krizhevsky, I. Sutskever and G. Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. NIPS*, pp. 1097-1105, 2012.

[11] P.-Y. Kuo, C.-Y. Wang and C.-Y. Huang. On rewiring and simplification for canonicity in threshold logic circuits. In *Proc. ICCAD*, pp. 396-403, 2011.

[12] R. Lippmann. An introduction to computing with neural nets. In *IEEE ASSP Magazine*, 4(2): 4-22, 1987.

[13] P. Merolla *et al.* A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197): 668-673, 2014.

[14] S. Muroga. *Threshold Logic and Its Applications*. John Wiley & Sons, 1971.

[15] A. Neutzling, J. Matos, A. Reis, R. Ribas and A. Mishchenko. Threshold logic synthesis based on cut pruning. In *Proc. ICCAD*, pp. 494-499, 2015.

[16] N. Nukala, N. Kulkarni and S. Vrudhula. Spintronic threshold logic array (STLA) – A compact, low leakage, non-volatile gate array architecture. *Journal of Parallel and Distributed Computing*, 74(6): 2452-2460, 2014.

Table 4: Comparison of CNF, AIG and PB-based equivalence verification (without iterative collapsing)

Statistics for equivalence verification between TLC before and after collapsing (w/o ite)									
	file generation time (s)			verification file size (KB)			verification time (s)		
circuit	CNF	CEC	PB	CNF	CEC	PB	CNF	CEC	PB
c3540	0.05	0.02	0.00	5304	504	112	44.58	0.35	1.76
c5315	0.07	0.03	0.00	7652	712	176	4.46	0.35	0.88
c6288	0.21	0.09	0.00	24252	2336	316	TO	2.03	TO
c7552	0.07	0.03	0.00	7636	764	208	2.71	0.44	1.89
s5378	0.07	0.03	0.00	7948	840	184	0.04	0.18	0.35
s9234.1	0.24	0.10	0.00	27732	2988	224	0.12	0.86	0.49
s13207	0.15	0.07	0.00	17492	2008	492	0.08	0.43	1.82
s15850	0.45	0.18	0.00	60716	4576	536	0.25	1.17	2.44
s35932	0.11	0.05	0.01	10276	1224	1524	18.78	1.15	25.20
s38417	0.57	0.24	0.01	72196	6360	1504	85.90	2.03	31.28
s38584	5.09	1.90	0.01	891944	76172	1512	3.57	22.66	28.97
b04	0.05	0.03	0.00	5740	836	72	13.62	1.61	0.12
b12	0.06	0.02	0.00	6556	720	160	3.47	0.16	0.24
b14	MO	MO	0.01	MO	MO	712	MO	MO	111.95
b15	MO	MO	0.01	MO	MO	1180	MO	MO	17.83
b17	MO	MO	0.03	MO	MO	4016	MO	MO	160.67
b18	MO	MO	0.09	MO	MO	11060	MO	MO	TO
b19	MO	MO	0.18	MO	MO	22368	MO	MO	TO
b20	MO	MO	0.01	MO	MO	1460	MO	MO	1795.46
b21	MO	MO	0.01	MO	MO	1492	MO	MO	2846.92
b22	MO	MO	0.02	MO	MO	2156	MO	MO	2524.79

[17] A. Palaniswamy and S. Tragoudas. An efficient heuristic to identify threshold logic functions. *ACM Journal on Emerging Technologies in Computing Systems*, 8(3): 19, 2012.

[18] A. Palaniswamy and S. Tragoudas. Improved threshold logic synthesis using implicant-implicit algorithms. *ACM Journal on Emerging Technologies in Computing Systems*, 10(3): 21, 2014.

[19] E. Sentovich *et al.* Sequential circuit design using synthesis and optimization. In *Proc. ICCD*, pp. 328-333, 1992.

[20] J. Subirats, J. Jerez and L. Franco. A new decomposition algorithm for threshold synthesis and generalization of Boolean functions. *IEEE Trans. on Circuits and Systems I: Regular Papers*, 55(10): 3188-3196, 2008.

[21] G. Tseitin. On the complexity of derivation in propositional calculus. In *Automation of Reasoning*, pp. 466-483, 1983.

[22] R. Zhang, P. Gupta, L. Zhong and N. Jha. Synthesis and optimization of threshold logic networks with application to nanotechnologies. In *Proc. DATE*, pp. 904-909, 2004.

[23] Y. Zheng, M. Hsiao and C. Huang. SAT-based equivalence checking of threshold logic designs for nanotechnologies. In *Proc. GLSVLSI*, pp. 225-230, 2008.

Table 5: Comparison of CNF, AIG and PB-based equivalence verification (with iterative collapsing)

Statistics for equivalence verification between TLC before and after iterative collapsing									
	file generation time (s)			verification file size (KB)			verification time (s)		
circuit	CNF	CEC	PB	CNF	CEC	PB	CNF	CEC	PB
c3540	0.04	0.01	0.00	3208	316	108	15.58	0.29	1.58
c5315	0.04	0.01	0.00	3052	324	176	1.52	0.29	1.51
c6288	0.12	0.04	0.01	9216	960	304	TO	1.60	TO
c7552	0.06	0.02	0.00	4960	512	208	1.55	0.39	2.18
s5378	0.10	0.03	0.00	7996	832	180	0.04	0.17	0.44
s9234.1	0.27	0.09	0.00	24340	2608	224	0.13	0.67	0.62
s13207	0.14	0.05	0.01	11728	1408	484	0.06	0.29	2.20
s15850	MO	2.84	0.01	MO	112484	532	MO	0.47	2.81
s35932	0.15	0.05	0.03	10276	1224	1524	15.83	1.16	18.66
s38417	1.81	0.55	0.03	190724	15248	1488	246.33	3.10	23.53
s38584	1.78	0.50	0.03	213720	18552	1500	0.81	4.97	26.33
b04	0.06	0.03	0.00	5716	832	72	8.30	1.66	0.12
b12	0.11	0.03	0.00	9356	1056	160	3.36	0.18	0.29
b14	1.05	0.27	0.01	111928	8336	672	1650.61	4.52	100.18
b15	0.80	0.25	0.02	79404	6740	1140	242.50	4.38	17.17
b17	MO	2.39	0.08	MO	93956	3676	MO	70.25	188.12
b18	MO	MO	0.24	MO	MO	10096	MO	MO	TO
b19	MO	MO	0.45	MO	MO	20464	MO	MO	TO
b20	3.38	0.99	0.03	393620	37684	1404	2506.10	17.81	1401.99
b21	2.19	0.61	0.03	245672	15728	1396	1240.13	7.87	1977.18
b22	5.84	1.58	0.05	668384	64828	2060	3634.66	39.72	3274.91