A new method to identify threshold logic functions

Seyed Nima Moazzafari, Spyros Tragoudas, Themistoklis Haniotakis
Department of Electrical and Computer Engineering
Southern Illinois University
Carbondale, USA
{nima.moazzafari, spyros, haniotak}@siu.edu

Abstract—An Integer Linear Programming based method to identify current mode threshold logic functions is presented. The approach minimizes the transistor count and benefits from a generalized definition of threshold logic functions. Process variations are taken into consideration. Experimental results show that many more functions can be implemented with predetermined hardware overhead, and the hardware requirement of a large percentage of existing threshold functions is reduced.

Keywords—threshold logic gate; synthesis; threshold function; current mode logic; weight assignment;

1. INTRODUCTION
Threshold Logic Gate is a promising candidate for the future digital circuits. Using TGs, circuits may be implemented with fewer number of logic gates, and hence with less delay, power dissipation, and silicon area [1-5]. A Boolean function that can be implemented as a single TG is called Threshold Logic Function (TF). A Boolean Threshold logic Function (TF) requires a weight, one per input variable. An n-input TF is realized using a sum of active weights where each active weight corresponds to a different input variable. For each input pattern, the function evaluates to one only when the sum of the active weights is greater than (or equal) to a weight value called the threshold weight [6]. In this paper, such a function is called a 1st-order TF and will be denoted as a 1-TF.

An n-input 1-TF \( f(x_1, x_2, \ldots, x_n) \) is formally defined as [6]:

\[
f(x_1, x_2, \ldots, x_n) = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} w_i \cdot x_i \geq w_T \\ 0 & \text{otherwise} \end{cases}
\]

where \( x_i \), \( i = 1, \ldots, n \), are binary input variables, \( w_i \) is the weight corresponding to the \( i \)th input, and \( w_T \) is the threshold weight (threshold value). In (1) the term \( w_i \cdot x_i \) indicates that the weight \( w_i \) is active when \( x_i = 1 \). The 1-TF \( f(x_1, x_2, \ldots, x_n) \) is also denoted by the set of weights \( \{w_1, w_2, \ldots, w_n\} \). For implementation purposes, the weights are assumed to be integers. A small fraction of binary functions are 1-TFs [6] and can be implemented as a single TG.

In a 2-TF, the weights are more than \( n \) and for each input pattern a weight can be activated either by a single input (as in 1-TF) and it is called a 1-weight, or by a pair of inputs and it is called a 2-weight. Likewise, \( k \)-TFs are defined when \( k > 2 \). However, the paper focuses on 2-TF due to space limitations. The 2-TF formulation for an \( n \)-input function is [7]

\[
f(x_1, x_2, \ldots, x_n) = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} w_i \cdot x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{i,j} \cdot x_i \cdot x_j \geq w_T \\ 0 & \text{otherwise} \end{cases}
\]

where \( x_i, i = 1, \ldots, n \), are binary input variables, \( w_i \) is the 1-weight corresponding to the \( i \)th input, \( w_{i,j} \) is the 2-weight corresponding to the pair of \( i \)th and \( j \)th inputs, and \( w_T \) is the threshold weight (threshold value). In (2) the term \( w_i \cdot x_i \) indicates that the weight \( w_i \) is active when \( x_i = 1 \) and the term \( w_{i,j} \cdot x_i \cdot x_j \) indicates that the weight \( w_{i,j} \) is active when \( x_i = 1 \) and \( x_j = 1 \). A 2-TF is also denoted by the set of weights \( \{w_1, w_2, \ldots, w_n, w_{1,2}, \ldots, w_{(n-1)n}; w_T\} \).

Several interesting circuit concepts have been proposed recently in [1, 4-5, 8-13], among others, for implementing TGs and synthesizing circuits using TGs. This paper focuses on the Current-mode TGs (CTGs) which is a popular PMOS- and NMOS-based implementation. (See [1, 9], among others.) however, the method is applicable to all existing implementations.

An automation framework is introduced to identify efficiently 2-TFs that can be implemented as current mode logic gates. A new method is proposed to implement CTG with minimum hardware overhead (transistor count) considering the definition in (2). The proposed identification method is an extension of the Integer Linear Programming (ILP) formulation in [6]. The objective function of the ILP is modified in order to control the number of 2-weights and minimize the area of the CTG. It will be shown that many more functions can be implemented as CTG using similar hardware overhead to that of traditional CTGs. Also, many 1-TFs are implemented as 2-TF with reduced hardware overhead. Process variations are also taken into consideration.

This paper is organized as follows. Section 2 shows how to implement current mode logic 2-TF with minimum hardware overhead in the presence of process variations. An ILP-based framework is also proposed to identify 2-TFs efficiently. Section 3 provides experimental results. Section 4 concludes the paper.

2. THE PROPOSED METHOD
The CTG implementation in [9] implements a 1-TF with NMOS (or PMOS) transistors connected in parallel. Let \( X \) denote the width of a minimum size transistor which implements the unit 1-weight. Each transistor implements a 1-weight \( w \) and its width is \( w \cdot X \), and the gate of the transistor is connected to an input. The gate of the NMOS transistor for the threshold is connected to the power supply (it is active for all input patterns). The length
of all transistors is the same and is determined by the used technology. We call such gates as 1-CTG.

A 2-weight is implemented with 2 NMOS (or PMOS) transistors of the same size (according to the respective weight) which are connected in series, and the transistor gates are connected to CTG inputs. The size of each transistor in a sub-circuit that implements a 2-weight with weight value \( w \) is set to \( 2 \cdot w \cdot X \) to keep the active current of a unit 2-weight equal to the active current of a unit 1-weight. The total area of a 2-weight sub-circuit is 4 times more than the area of a 1-weight that implements the same weight. This area ratio is called the penalty factor and is taken into consideration to force the ILP-solver to find the minimum possible transistor count which is the total number of unit size transistors that implements weights. (In CTG, the area of a transistor with twice the unit width is practically the same as two minimum size transistor.) Such gates are called 2-CTGs.

An ILP formulation is presented to implement a Unate Function (UF) as a 2-CTG, as in (2) with minimum transistor count. A function is a UF if it is either positive or negative in every variable \([6]\). A function is positive (negative) in variable \( x_i \) if the variable \( x_i (x_i) \) does not appear in the expression of the function. The proposed ILP is an improvement of \([7]\).

The ILP contains \( 2^n + n \) constraints with \( n + n' + 1 \) variables, where \( n' = \binom{n}{2} \) is the total number of 2-weights. There is a constraint per input pattern to satisfy the functionality, and \( n \) constraints that bind the range of each weight. Once a UF \( f \) is given, the Modified Chow’s parameters \([6]\) of all inputs and pairs of inputs determine the negative weights (including 1-weights and 2-weights). To form an efficient ILP, every input \( x_i \) (or product of two inputs \( x_i \cdot x_j \)), that activates a 1-weight (or 2-weight) with a negative Modified Chow’s parameter must be complemented. A 1-weight with negative Modified Chow’s parameter is activated when at least one of \( x_i \) and \( x_j \) is set to 0. The ILP with the appropriate activation signals determines whether function \( f \) is a 2-CTF.

Furthermore, the penalty factors appear as the coefficient of weights in the objective function of the ILP. The following objective function minimizes the CTG transistor count:

\[
\text{minimize: } w_T + \sum_{i=1}^{n} w_i + 4 \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{ij}
\]

(3)

The weight configuration will then be assigned using the following property which is called the negation property: The negation of variable \( x_i \) \((x_i \rightarrow \bar{x}_i)\), changes the weight configuration to \([w_1, w_2, \ldots, -w_j, \ldots, w_i; w_T - w_i] \). The following illustrates the ILP-based method to identify a 2-CTF.

**Example 1:** Consider a 4-input UF \( F_1 = x'_1 \cdot x_2 \cdot x'_3 \cdot x_4 \) with a set of all unknown weights \( w = [w_1, w_2, w_3, w_4, w_{1,2}, w_{1,3}, w_{1,4}, w_{2,3}, w_{2,4}, w_{3,4}; w_T] \). The set of Modified Chow’s parameters of either an input or a pair of inputs that activates a weight is \( m_{F_1} = (-5, -1, +3, -1, -9, -5, -9, -5, -7, -5) \). To form an efficient ILP, every activation signal with negative \( m_i \) must be complemented. In this example, all inputs and pairs of inputs must be complemented except \( x_3 \).

**Example 2:** Consider the 4-variable function \( F_2 = x_4 \cdot x_3 \cdot x_2 \cdot x_1 \). It is a 1-CTF with optimum weight configuration \( w = [w_1, w_2, w_3, w_4; w_T] = [4, 4, 6, 2, 7] \) using the ILP in \([6]\). The transistor count or the total sum of threshold and input weights of 1-CTG is 23. Let \( X \) denote the area of a unit 1-weight transistor. The total area is \( 23 \cdot X \).

However, this function can be implemented as a 2-CTG with weight configuration \( w = [w_1, w_2, w_3, w_4, w_{1,2}, w_{1,3}, w_{1,4}, w_{2,3}, w_{2,4}, w_{3,4}; w_T] = [2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 2, 3] \). Each 2-weight requires 4 more times the transistor count of a 1-weight that implements the same weight. Thus, the total area of the 2-CTG reduces to \( X \cdot (2 + 2 + 2 + X) \cdot (2 + X) \cdot (3) \). Fig. 1 shows the 1-CTG \([9]\) and proposed 2-CTG implementations of \( F_2 \) and the size of transistors that implements 1-weights and 2-weights considering \( X \) as the size of a minimum width transistor to implement a unit 1-weight. The length of all the PMOS and NMOS transistors is the same and determined by the used technology.

The correlation between the sign of the Modified Chow’s parameters and the sign of weights only holds for UF's. Nonunate functions are called Binate Functions (BFs). The ILP for a BF works as in \([7]\) and contains \( 2^n + 3n \) constraints with \( 3(n + n' + 1) \) variables, where \( n' = \binom{n}{2} \) is the total number of 2-
weights. Each weight can be either positive or negative. The objective function is to minimize quantity \(|w_p| + 1 \cdot \sum_{i=1}^{n} |w_i| + 4 \cdot \sum_{j=1}^{n-1} |w_i|\), where \(|w_p|\), denotes the absolute value for each weight \(w_p\), and \(w_i \in \{w_{r1}, w_{r2}, w_{l1}\}\). Let \(y_x\) be a binary variable, and \(U\) denote a predetermined upper bound of \(|w_p|\) for each weight \(w_p\). For each \(w_p\), two variables \(w_p^+\) and \(w_p^-\) are used, and the bound on the absolute value \(w_p\) is enforced using the following constraints:

\[
\begin{align*}
0 \leq w_p^+ & \leq U \cdot y_p \\
0 \leq w_p^- & \leq U \cdot (1 - y_p) \\
y_p & \in \{0, 1\}
\end{align*}
\]

Then \(w_p = w_p^+ - w_p^-.\) In addition, for CTG the ILP should minimize quantity

\[
w_r^+ + w_r^- + \sum_{l=1}^{n} (w_l^+ + w_l^-) + 4 \cdot \sum_{i=1}^{n-1} (w_i^+ + w_i^-)
\]

Then \(w_p = w_p^+ - w_p^-\). In addition, for CTG the ILP should minimize quantity

\[
3. EXPERIMENTAL RESULTS

The proposed ILP-based approaches have been developed in the C++ language on an Intel Xenon 2.4GHz with 8GB memory. To evaluate the contribution, we examined up to eighteen input non-scalable functions. (An \(n\)-input non-scalable function requires no less than \(n\) non-empty levels of variables in the BDD representation for some ordering of the variables \([6]\).) The weight values were integers in the range \([-25, 25]\).

Table 2 reveals the number of TFs that are implemented as 2-CTGs but have no more transistor count, the number of unit size transistors that implement weights, than the maximum transistor count of any examined non-scalable \(n\)-input 1-TF implemented as 1-CTG. Results for up to eighteen inputs were derived using the robust ILP-based approach considering weight variations.

Penalty factors 1 and 4 are also applied to the objective function of ILP for 1-weights and 2-weights, respectively. Column one of Table 2 lists the number of inputs. The second column lists the number of 1-TFs that can be implemented as a single 1-CTG. For functions with greater than four inputs, the entries in Table 2 are obtained by sampling randomly 20,000 functions. The \(2^n\) bits of input vector of an \(n\)-input function is filled with one bit values at positions determined by randomly selecting an integer mod \(2^n\), and the number of ones in the function must obey the distribution of functions based on this property. (For example, the number of 5-input functions with 16 ones in the output bit vector is approximately 10 times more than the number of 5-input functions with 10 ones.) In order for the experiment to have more statistical significance, we only consider non-scalable functions, and when a function is generated we apply the procedure described earlier in this section to determine that it is non-scalable. (It is asserted that the distribution of non-scalable \(n\)-input functions based on the number of ones in their output bit vector is the same as the one described earlier for \(n\)-input functions.)

The third column shows the number of 2-TFs that do not have 2-CTG transistor count (area) higher than any of 1-TFs in column two, and the fourth column their increase on the number of TFs which were determined by the number of 2-TFs over the number of 1-TFs. Columns four to seven show similar results when considering 6% and 12% weight variations. These values obtained by SPICE simulations in corner cases using 45nm technology \([14]\) while considering 5% and 10% variations in transistor width and length \([15]\). These results show that a significant percentage of functions can be implemented in current mode gates using transistor count similar to that for the very few 1-TF functions that implemented as 1-CTG, and in the presence of process variations.

Table 3 shows that many 1-TF functions can be identified as 2-TF with less area in the CTG implementation. The first row lists the values of \(n\), the number of input variables. The second row lists the number of non-scalable 1-TF that were identified for each value of \(n\). The third row shows the number of 1-TF which were also identified as 2-TF with less area in the CTG implementation.

---

**Fig. 1. The CTG implementation for function**

\[F_2 = x_4 x_3 + x_2 x_4 + x_4 x_1\] with (a) 1-CTG as 1-TF \([9]\) (b) proposed 2-CTG as 2-TF.

---

<table>
<thead>
<tr>
<th>(n)</th>
<th># of 1-CTG</th>
<th># of 2-CTG</th>
<th>Increase ratio</th>
<th># of 2-CTG</th>
<th>Increase ratio</th>
<th># of 2-CTG</th>
<th>Increase ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>8</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>72</td>
<td>72</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1536</td>
<td>2742</td>
<td>1.8</td>
<td>1.75</td>
<td>1.55</td>
<td>1.55</td>
<td>1.55</td>
</tr>
<tr>
<td>5</td>
<td>99</td>
<td>1346</td>
<td>13.6</td>
<td>13.6</td>
<td>13.3</td>
<td>13.3</td>
<td>13.3</td>
</tr>
<tr>
<td>6</td>
<td>120</td>
<td>1859</td>
<td>15.5</td>
<td>15.5</td>
<td>13.9</td>
<td>13.9</td>
<td>13.9</td>
</tr>
<tr>
<td>7</td>
<td>84</td>
<td>1804</td>
<td>21.5</td>
<td>21.5</td>
<td>20.4</td>
<td>20.4</td>
<td>20.4</td>
</tr>
<tr>
<td>8</td>
<td>76</td>
<td>2216</td>
<td>29.1</td>
<td>29.1</td>
<td>27.6</td>
<td>27.6</td>
<td>27.6</td>
</tr>
<tr>
<td>9</td>
<td>163</td>
<td>1462</td>
<td>8.9</td>
<td>8.9</td>
<td>8.4</td>
<td>8.4</td>
<td>8.4</td>
</tr>
<tr>
<td>10</td>
<td>137</td>
<td>2047</td>
<td>14.9</td>
<td>14.9</td>
<td>14.3</td>
<td>14.3</td>
<td>14.3</td>
</tr>
<tr>
<td>11</td>
<td>85</td>
<td>1887</td>
<td>22.2</td>
<td>22.2</td>
<td>19.4</td>
<td>19.4</td>
<td>19.4</td>
</tr>
<tr>
<td>12</td>
<td>85</td>
<td>1086</td>
<td>12.7</td>
<td>12.7</td>
<td>12.2</td>
<td>12.2</td>
<td>12.2</td>
</tr>
<tr>
<td>13</td>
<td>97</td>
<td>1815</td>
<td>18.7</td>
<td>18.7</td>
<td>14.4</td>
<td>14.4</td>
<td>14.4</td>
</tr>
<tr>
<td>14</td>
<td>64</td>
<td>1332</td>
<td>20.8</td>
<td>20.8</td>
<td>18.4</td>
<td>18.4</td>
<td>18.4</td>
</tr>
<tr>
<td>15</td>
<td>83</td>
<td>1492</td>
<td>17.9</td>
<td>17.9</td>
<td>16.4</td>
<td>16.4</td>
<td>16.4</td>
</tr>
<tr>
<td>16</td>
<td>72</td>
<td>1122</td>
<td>15.6</td>
<td>15.6</td>
<td>14.0</td>
<td>14.0</td>
<td>14.0</td>
</tr>
<tr>
<td>17</td>
<td>63</td>
<td>1247</td>
<td>19.8</td>
<td>19.8</td>
<td>17.5</td>
<td>17.5</td>
<td>17.5</td>
</tr>
<tr>
<td>18</td>
<td>60</td>
<td>1098</td>
<td>18.3</td>
<td>18.3</td>
<td>15.5</td>
<td>15.5</td>
<td>15.5</td>
</tr>
</tbody>
</table>
the same as for the 1-CTGs.

4. CONCLUSIONS

It has been demonstrated that the presented approach can implement many more functions as current mode gates with less area, similar power dissipation, and slightly increased delay when compared to existing method. In future works, heuristic approaches can be used to implement 2-TFs with higher inputs by extending a work.

ACKNOWLEDGMENT

This research has been supported in part by grants NSF IIP 1432026, and NSF IIP 1361847 from the NSF I/UCRC for Embedded Systems at SIUC. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES