LUT Cascade Realization of Threshold Functions and Its Application to Implementation of Ternary Weight Neural Networks

Tsutomu Sasao
Department of Computer Science
Meiji University, Kawasaki, 214-8571, Japan

***Abstract*—This paper shows a method to realize a threshold logic function using K-LUTs. It considers two cases: 1) when the function has arbitrary weights and 2) when the number of distinct non-zero weights is fixed to r. In the latter case, this paper shows that any n-variable threshold logic function can be implemented with $O(n^{r+1}\log n)$ K-LUTs in a form of an LUT cascade. It also shows that any n-variable threshold logic function with weights $(-1,0,1)$ or $(-2,-1,0,1,2)$ can be implemented with $O(n^2\log n)$ K-LUTs.**

***Keywords*-threshold function, cascade realization, ternary weight neural network, functional decomposition, binary decision diagram**

I. Introduction

Neural networks have wide applications in various areas, including image recognition. For applications that require high throughput, FPGAs are often used to implement them [3], [20]. Since neural networks consist of millions of threshold gates, efficient realizations of threshold gates by K-LUTs are desired, where K denotes the number of inputs to an LUT. In commercial FPGAs, LUTs with $K = 6$ are available.

To estimate the number of LUTs necessary to implement a given threshold gate of n variables, Shannon expansion is often used [20]. In such a method, the number of necessary LUTs increases exponentially with n. When the weights of the threshold gates are small, threshold logic functions can be efficiently implemented by LUT cascades [15], [17]. However, when the weights of a threshold logic function are large, the upper bound given in [15] is not useful. In fact, there exists a series of threshold logic functions, whose weights increase exponentially with n.

When neural networks are designed for FPGAs, the number of distinct weights of threshold gates is fixed. That is, r, the number of distinct weights of the threshold gates, is independent of the number of the input variables n. In these cases, we assume that activations, i.e., input variables, are two-valued.

In this paper, we consider realizations of threshold logic functions for various cases. We show that, when the number of distinct weights of the threshold logic function f is fixed to r, the number of K-LUTs to realize f is $O(n^{r+1}\log n)$. Especially, any n-variable threshold function with weights $(-1,0,1)$ or $(-2,-1,0,1,2)$ can be realized with $O(n^2\log n)$ K-LUTs.

The rest of this paper is organized as follows: Section II reviews basic theories used in this paper. Section III introduces threshold logic functions, and considers the column multiplicity of the decomposition chart. Section IV shows LUT cascade realizations of threshold logic functions. Section V considers the case for totally symmetric threshold logic functions. Section VI considers the case for threshold logic functions where the number of distinct weights is fixed. Section VII considers the application of the presented results to ternary weight neural networks. Section VIII reviews related works. Section IX concludes the paper.

II. Basic Properties

In this section, we review LUT cascade synthesis [17] of a binary logic function $B^n \to B$, where $B = \{0,1\}$.

Definition 2.1 ([2]): Let $f(X)$ be a logic function, and (X_1, X_2) be a partition of the input variables X, where $X_1 = (x_1, x_2, \ldots, x_k)$ and $X_2 = (x_{k+1}, x_{k+2}, \ldots, x_n)$. The **decomposition chart** for f is a two-dimensional matrix with 2^k columns and 2^{n-k} rows, where each column and row is labeled by a unique binary code, and each element corresponds to the truth value of f. The function represented by a column is a **column function** and depends on X_2. Variables in X_1 are **bound variables**, while variables in X_2 are **free variables**. In the decomposition chart, the **column multiplicity**, denoted by μ_k, is the number of different column functions.

Theorem 2.1 ([4]): For a given logic function f, let X_1 be the bound variables, let X_2 be the free variables, and let μ_k be the column multiplicity of the decomposition chart. Then, the function f can be realized with the network shown in Fig. 2.1. In this case, the number of signal lines connecting blocks H and G is $\lceil \log_2 \mu_k \rceil$.

Theorem 2.2 ([2]): Let $\mu_k(n)$ be the column multiplicity of a decomposition chart of an n-variable logic function with k bound variables. Then,

$$\mu_k(n) \le \min\{2^k, 2^{2^{n-k}}\}.$$

When circuits are designed by LUTs, functions with smaller column multiplicities tend to have smaller realizations.

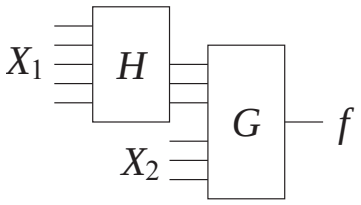

Fig. 2.1. Realization of a logic function by decomposition.

Definition 2.2: Let $f(x_1, x_2, \ldots, x_n)$ be a logic function. The **profile** of the function f is the vector $(\mu_1, \mu_2, \ldots, \mu_n)$, where μ_k denotes the column multiplicity of the decomposition chart for $f(X_1, X_2)$, $X_1 = (x_1, x_2, \ldots, x_k)$ and $X_2 = (x_{k+1}, \ldots, x_n)$, assuming that the order of variables $(x_1, x_2, \ldots, x_n)$ is fixed. The **C-measure** of the function f is $\max(\mu_1, \mu_2, \ldots, \mu_n)$, and is denoted by $\mu(f)$.

The order of the variables affects the C-measure, but we choose the **natural order** $(x_1, x_2, \ldots, x_n)$ of the input variables. C-measures and profile can be efficiently obtained by widths of a quasi-reduced ordered BDD [17]. A logic function with a small C-measure can be realized by a compact LUT cascade.

Corollary 2.1: Let f be an arbitrary n-variable logic function. Then,

$$\mu(f) \leq \max_{k=1}^{n} \left[\min\{2^k, 2^{2^{n-k}}\} \right].$$

For any partition (X_1, X_2) of X, we have the decomposed realization shown in Fig. 2.1. By repeatedly applying functional decompositions to a given logic function $f(X) = f(X_1, X_2, \ldots, X_s)$, we have an **LUT cascade** [17] shown in Fig. 2.2. An LUT cascade consists of **cells**. The signal lines connecting adjacent cells are **rails**.

Lemma 2.1 ([17]): An arbitrary logic function f can be realized by an LUT cascade, whose cells have at most $\lceil \log_2 \mu(f) \rceil + 1$ inputs, and at most $\lceil \log_2 \mu(f) \rceil$ outputs, where $\mu(f)$ is the C-measure of f.

Lemma 2.2 ([15]): In an LUT cascade that realizes an n-variable logic function f, let s be the number of cells; K be the number of inputs to a cell; $K \geq R + 1$, where $R = \lceil \log_2 \mu(f) \rceil$; and $n \geq K + 1$. Then, an LUT cascade satisfying the following condition exists:

$$s = \left\lceil \frac{n - R}{K - R} \right\rceil.$$

Example 2.1: Consider the function $f(X_1, X_2)$ whose decomposition chart is shown in Fig. 2.3, where blank entries denote 0's. In this case, $X = (x_1, x_2, x_3, x_4, x_5, x_6)$ is partitioned into $X_1 = (x_1, x_2, x_3)$ and $X_2 = (x_4, x_5, x_6)$. The column multiplicity of Fig. 2.3 is $\mu_3 = 4$, since four distinct column functions exist. The profile of the function is (2,3,4,3,3,2). The decomposition chart whose bound variables are $\{x_1, x_2, x_3, x_4\}$ is shown in Fig. 6.2. And the C-measure of the function is $\mu(f) = 4$. Since $R = \lceil \log_2 \mu(f) \rceil = 2$, by Lemma 2.2, f can be realized by an LUT cascade, where each LUT has at most $K = R + 1 = 3$ inputs. The number of cells is at most

$$s = \left\lceil \frac{n - R}{K - R} \right\rceil = \frac{6 - 2}{3 - 2} = 4.$$

■

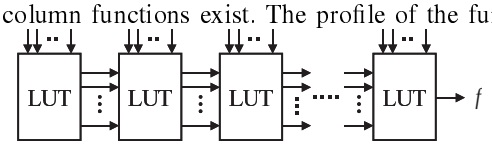

Fig. 2.2. LUT cascade.

			0	0	0	0	1	1	1	1	x_3
			0	0	1	1	0	0	1	1	x_2
			0	1	0	1	0	1	0	1	x_1
0	0	0									
0	0	1								1	
0	1	0						1	1	1	
0	1	1		1	1	1	1	1	1	1	
1	0	0						1	1	1	
1	0	1		1	1	1	1	1	1	1	
1	1	0	1	1	1	1	1	1	1	1	
1	1	1	1	1	1	1	1	1	1	1	
x_6	x_5	x_4									

Fig. 2.3. Decomposition chart for $f(X_1, X_2)$, where $X_1 = (x_1, x_2, x_3)$ and $X_2 = (x_4, x_5, X_6)$.

When the column multiplicity is greater than 32, the number of rail outputs is more than five. In such a case, we synthesize K-input LUTs by 6-LUTs.

Lemma 2.3: An arbitrary n-variable logic function can be represented as follows:

$$f(X_1, X_2) = \bigvee_{i \in P} g_i(X_1) X_2^i,$$

where $X_1 = (x_1, x_2, \ldots, x_k)$, and $X_2 = (x_{k+1}, x_{k+2}, \ldots, x_n)$, $P = \{0, 1, \ldots, 2^{n-k} - 1\}$, and the OR is performed with respect to 2^{n-k} elements.

Theorem 2.3 ([17]): The number of 6-LUTs to realize an arbitrary n-variable logic function $(n \geq 6)$ f is at most

- $(2^{n-4} - 1)/3$, when n is even, and
- $(2^{n-4} + 1)/3$, when n is odd.

III. Threshold Logic Functions

In this section, we show some properties of threshold logic functions, which are useful for LUT cascade synthesis.

Definition 3.1: A **threshold logic function** $f(x_1, x_2, \ldots, x_n)$ satisfies the relation: $f = 1$ if $\sum_{i=1}^{n} w_i x_i \geq T$, and $f = 0$ if $\sum_{i=1}^{n} w_i x_i \leq T - 1$, where $(w_1, w_2, \ldots, w_n)$ are **weights**, and T is an integer called **threshold**. Any threshold logic function can be represented by a **structure of the threshold function**:

$$(w_1, w_2, \ldots, w_n; T).$$

Definition 3.2: A function f is positive in x_i if f can be represented as $f = f_0 \vee x_i f_1$, where f_0 and f_1 are functions

that are independent of x_i. f is **positive** if f is positive for all the variables in f. Similarly, a function f is **negative** in x_i if f can be represented as $f = f_0 \vee \bar{x}_i f_1$.

In a threshold logic function f, if the weight w_i is negative, then the function is negative in x_i. In such a case, the function can be converted into positive by replacing the negative variable x_i with $1 - y_i$.

Thus, without loss of generality, we can assume that in the given function, weights and the threshold are positive numbers. However, theorems hold even if weights and/or threshold are negative.

Definition 3.3 ([14]): An **optimal structure** of an n-variable positive threshold logic function has the minimum value of $\sum_{i=1}^{n} |w_i|$, where w_i are the weights of the function.

An optimal structure of a threshold logic function can be found by a linear integer programming.

Theorem 3.1: Consider a decomposition chart of a threshold logic function. Let $X_1 = (x_1, x_2, \ldots, x_k)$ be bound variables, and $(w_1, w_2, \ldots, w_k)$ be their weights. Then, the column multiplicity of f is at most $UB1 = 1 + \sum_{i=1}^{k} |w_i|$.

(Proof)[1] Let f be a positive function. By Definition 3.1, the value of the threshold logic function depends on the value of

$$WS = \sum_{i=1}^{k} w_i x_i.$$

There are $\binom{n}{k}$ different ways to select bound variables. When the variables in the bound set are fixed, WS takes values between 0 and $WS1 = \sum_{i=1}^{k} w_i$. Thus, the number of different column functions is at most $1 + WS1$. □

Theorem 3.2: Consider the decomposition chart of a threshold function $f(X_1, X_2)$, where X_1 are the bound variables, and $X_2 = (x_{k+1}, x_{k+2}, \ldots, x_n)$ are the free variables. Let the weights for the free variables be $(w_{k+1}, w_{k+2}, \ldots, w_n)$. Then, the column multiplicity of the decomposition chart is at most $UB2 = 2 + \sum_{i=k+1}^{n} |w_i|$.

(Proof) Assume that the function is positive. The column functions are threshold logic functions with weights $(w_{k+1}, w_{k+2}, \ldots, w_n)$, where different coelum functions may have different thresholds. Thus, the column multiplicity is at most the number of different thresholds. Thresholds T of such functions are between 1 and $WS2 = \sum_{i=k+1}^{n} w_i$. Also, there is the constant 1 function when $T = 0$, and the constant 0 function when $T = WS2 + 1$. Hence, the theorem. □

Example 3.1: The decomposition chart in Fig. 2.3 shows the threshold logic function with the structure (1,1,2,2,3,3;6). Thus, f can be written as

$$f = \begin{cases} 1 & (x_1 + x_2 + 2x_3 + 2x_4 + 3x_5 + 3x_6 \geq 6) \\ 0 & \text{(Otherwise).} \end{cases}$$

By Theorem 3.1, the column multiplicity of Fig. 2.3 is at most

$$UB1 = 1 + (1 + 1 + 2) = 5.$$

On the other hand, when $n = 6$ and $k = 3$, the upper bound on the column multiplicity of an arbitrary logic function obtained by Theorem 2.2 is $min\{2^k, 2^{2^{n-k}}\} = min\{2^3, 2^8\} = 8$. Thus, the knowledge of the structure of threshold logic function is useful to improve the upper bound. ■

Definition 3.4: Let $\theta(n)$ be the number of positive threshold logic functions with up to n variables,

TABLE 3.1
NUMBER OF FUNCTIONS WITH n OR FEWER VARIABLES.

Number of Variables	Logic Functions	Threshold Functions	Positive Functions	Positive Threshold Functions
0	2	2	2	2
1	4	4	3	3
2	16	14	6	6
3	256	104	20	20
4	65,536	1,882	168	150
5	4,294,967,296	94,572	7,581	3,287

Table 3.1 compares the numbers of various classes of functions. The first column shows the number of variables n; the second column shows the number of logic functions with n or fewer variables, that is 2^{2^n}; the third column shows the number of threshold logic functions with n or fewer variables [14]; the fourth column shows the number of positive functions with n or fewer variables, which is equal to the Dedekind number [18]; the last column shows $\theta(n)$, the number of positive threshold logic functions with n or fewer variables, which is equal to the number of N-equivalence classes of threshold logic functions with n or fewer variables [14].

Theorem 3.3: Consider a decomposition chart of an n-variable positive threshold logic function f. Let k be the number of bound variables. The column multiplicity of an n-variable positive threshold logic function f is at most

$$min\{2^k, \theta(n-k)\}.$$

(Proof) In a decomposition chart, each column denotes a positive threshold logic function of $n - k$ variables. The number of columns is 2^k. The number of different column functions is at most $\theta(n-k)$. Hence, we have the theorem. □

Example 3.2: Consider the function f that appeared in Example 3.1. It is a positive threshold logic function with $n = 6$ variables. Consider the decomposition chart where $k = 4$. Then, by Theorem 3.3, the column multiplicity is at most $min\{2^k, \theta(n-k)\} = min\{2^4, 6\} = 6$. Fig.6.2 is the decomposition chart. The column multiplicity is three. ■

Theorem 3.4: Consider a decomposition chart of a positive threshold logic function f with a threshold T, then the column multiplicity is at most $T + 2$.

(Proof) In a decomposition chart of f, columns represent positive threshold logic functions sharing the same weights. The number of such functions is at most $T+2$, if we consider two constant functions. □

Note that the LUT cascade shown in Fig. 2.2 realizes any threshold logic function with the weights $(w_1, w_2, \ldots, w_n)$, but different threshold T. To change the threshold T, only the content of cells must be modified.

[1]This proof is simpler than one in [15].

IV. Threshold Functions with Arbitrary Weights

Lemma 4.1: Any threshold logic function with $n = 8$ variables can be realized with at most four 6-LUTs.

(Proof) Consider the decomposition chart of a threshold logic function $f(X_1, X_2)$, where $X_1 = (x_1, x_2, \ldots, x_6)$ and $X_2 = (x_7, x_8)$. By Theorem 3.3, the column multiplicity is at most $\theta(2) = 6$. Thus, the function f can be implemented by the circuit shown in Fig. 4.1. Note that the first cell has six external inputs X_1 and produces 3 rail outputs, since $\lceil \log_2 \theta(2) \rceil = 3$. The number in the cell shows the number of 6-LUTs to realize the cell. Thus, the total number of 6-LUTs to implement the function is at most four. □

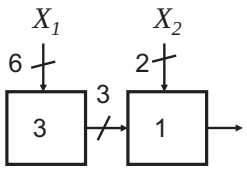

Fig. 4.1. Realization of a threshold function of 8-variables.

Note that to realize a non-threshold logic function of 8 variables, we need at most five 6-LUTs, by Theorem 2.3.

V. Threshold Functions with Unit Weights

In this section, we consider the number of LUTs to realize a threshold logic function with unit weights.

Definition 5.1: [16] A function f is **totally symmetric** if any permutation of the variables in f does not change the function.

In the case of binary weight neural networks, the weights can be either 0 or 1. However, when the weight for x_i is 0, the function is independent of x_i. Thus, only positive symmetric threshold logic functions need to be considered.

Lemma 5.1: The number of distinct positive non-zero totally symmetric threshold logic functions of n variables is $n + 1$.

(Proof) The function value depends only on the number of 1's in the input variables. □

Example 5.1: When $n = 3$, $n + 1 = 4$ different non-zero positive symmetric threshold logic functions exist:

$$x_1x_2x_3, \qquad x_1x_2 \vee x_2x_3 \vee x_3x_1,$$
$$x_1 \vee x_2 \vee x_3, \qquad 1$$

■

Lemma 5.2: Let f be a positive totally symmetric threshold logic function of n variables. Then,

$$\mu(f) = \max_{k=1}^{n} \Big[\min\{k+1, n-k+2\} \Big].$$

(Proof) Consider the decomposition chart of f, where $X_1 = (x_1, x_2, \ldots, x_k)$ denotes the bound variables and $X_2 = (x_{k+1}, x_{k+2}, \ldots, x_n)$ denotes the free variables.

By Theorem 3.1, the number of distinct column functions is at most $k+1$. Also, by Theorem 3.2, the number of distinct column functions is at most $n - k + 2$, since the column functions are positive symmetric threshold logic functions of $n - k$ variables, and two constant functions. □

Theorem 5.1: Let f be a positive totally symmetric threshold logic function f with n variables. Then, $\mu(f) \leq \left\lceil \frac{n}{2} \right\rceil + 1$.

(Proof) When n is an odd number: By Lemma 5.2, $\min\{k+1, n-k+2\}$ takes its maximum when $k+1 = n-k+2$. In this case, $k+1 = \lceil \frac{n}{2} \rceil + 1$.

When n is an even number: By Lemma 5.2, when $k = \lceil \frac{n}{2} \rceil$, $\min\{k+1, n-k+2\}$ takes its maximum $k+1 = \lceil \frac{n}{2} \rceil + 1$. □

Example 5.2: Consider a positive symmetric threshold logic function f of $n = 12$ variables. From Theorem 5.1, the column multiplicity is at most $\left\lceil \frac{12}{2} \right\rceil + 1 = 6 + 1 = 7$. Thus, f can be implemented by the LUT cascade shown in Fig. 5.1. Note that the number of 6-LUTs is 7. When the threshold is 7, the profile of the function f is
$(2, 3, 4, 5, 6, 7, 7, 6, 5, 4, 3, 2)$. ■

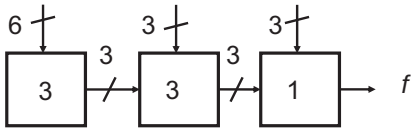

Fig. 5.1. Realization of a totally symmetric threshold function of 12-variables.

Theorem 5.2: A totally symmetric threshold logic function f of n variables can be realized using the following number of 6-LUTs:

- $n + 1$ (when $n \leq 21$).
- $2n - 19$ (when $22 \leq n \leq 31$)
- $5n - 113$ (when $32 \leq n \leq 62$).

(Proof) These numbers of 6-LUTs were obtained by computers. □

VI. Threshold Functions with Fixed Number of Distinct Weights

Definition 6.1: In a logic function $f(X_1, X_2, \ldots, X_r)$, if any permutation of variables in X_i, where $i = 1, 2, \ldots, r$, do not change the function, then f is **partially symmetric** with respect to X_i.

Theorem 6.1: The number of distinct partially symmetric positive threshold logic functions $f(X_1, X_2, \ldots, X_r)$ is at most $2 + \sum_{i=1}^{r} |w_i| n_i$, where n_i is the number of variables in X_i, and w_i is the weight of variables in X_i.

(Proof) Let $WS = \sum_{i=1}^{r} w_i n_i$. Then, the threshold can take values between 0 and WS. So, we have at most $WS + 1$ different functions. In addition, constant 0 can be the function. □

Example 6.1: Consider a set of four-variable partially symmetric positive threshold logic functions $f(X_1, X_2)$, where $X_1 = (x_1, x_2)$ and $X_2 = (x_3, x_4)$. Let the weights for variables in X_1 and X_2 be 1 and 2, respectively. In this case, the function can be written as:

$$f(X_1, X_2) = 1 \Leftrightarrow WS = (x_1 + x_2) + 2(x_3 + x_4) \geq T.$$

In addition, there are constant 0 and constant 1 functions. Thus, by Theorem 6.1, there are at most $2+1\times 2+2\times 2=8$ distinct functions. ■

Example 6.2: Let $f(X_1, X_2, X_3)$ be an 18-variable threshold function, where $X_1 = (x_1, x_2, \ldots, x_6)$, $X_2 = (x_7, x_8, \ldots, x_{12})$, and $X_3 = (x_{13}, x_{14}, \ldots, x_{18})$. Let the weights of the variables X_1, X_2 and X_3 be 1, 2, and 3, respectively. Then, the function can be realized by the cascade shown in Fig. 6.1.

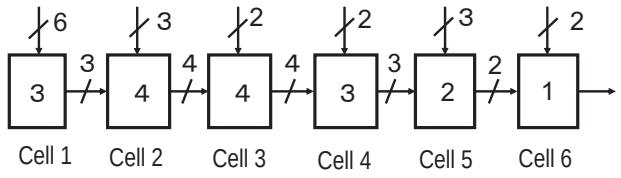

Fig. 6.1. Realization of threshold functions of 18-variables.

- In Cell 1, the number of external inputs is 6, and the number of rail outputs is 3, since the number of different column functions is $1+\sum_{i=1}^{6} w_i = 7$, by Theorem 3.1.
- In Cell 2, the number of external inputs is 3, and the number of rail input is 3. By Theorem 3.1, the number of different column functions is at most

$$1+\sum_{i=1}^{6} w_i + \sum_{j=7}^{9} w_j = 13.$$

  By Theorem 2.1, the number of rail output is 4.
- In Cell 3, the number of external inputs is 2, and the number of rail input is 4.
  Consider the weights of remaining variables. The weight of x_{12} is 2, but the weights of remaining 6 variables $x_{13}, x_{14}, \ldots, x_{18}$ are 3. So, the column functions are partially symmetric. Note that

$$WS = 2x_{12} + 3(x_{12} + x_{13} + \cdots + x_{18})$$

  takes at most $(1+1)\times(6+1) = 14$ distinct values. As will be shown in Theorem 6.2, the number of distinct column functions can be at most $1+14=15$. Thus, the number of rail outputs of Cell 3 is 4.
- In Cell 4, the number of external inputs is 2, and the number of rail inputs is 4. Since the weights of remaining variables are all 3, the columns represent totally symmetric function of 5 variables. By Lemma 5.1, the number of such functions is at most 7, +1 is for the constant 0 function. Thus, the number of rail outputs of Cell 4 is 3.
- In Cell 5, the number of external inputs is 3, and the number of rail input is 3. Since the weights of remaining variables are all 3, the columns represent totally symmetric function of 2 variables. Since, the number of such functions is at most 4, the number of rail outputs of Cell 5 is 2.
- In Cell 6, the number of external inputs is 2, and the number of rail input is 2.

The total number of 6-LUTs in Fig.6.1 is 17. When the threshold of the function f is $T = 17$, the profile of the function is $(2, 3, 4, 5, 6, 7, 9, 11, 13, 15, 11, 7, 7, 6, 5, 4, 3, 2)$. ■

Theorem 6.2: Consider a partially symmetric positive threshold logic function. If the non-zero weights w_i of free variables have r distinct values, then the number of distinct column functions is at most

$$1+\prod_{i=1}^{r}(n_i+1),$$

where n_i is the number of free variables with weights w_i.

(Proof) By Lemma 5.1, we have the theorem. +1 is for the constant 0 function. □

Example 6.3: The function f shown in Fig.2.3 is a partially symmetric positive threshold logic function, and can be written as $f(Y_1, Y_2, Y_3)$, where $Y_1 = (x_1, x_2), Y_2 = (x_3, x_4)$ and $Y_3 = (x_5, x_6)$. Fig. 6.2 is the decomposition chart of $f(Z_1, Z_2)$, where $Z_1 = (x_1, x_2, x_3, x_4)$ and $Z_2 = (x_5, x_6)$. Note that the column multiplicity is three. Non-zero weights of the free variables (i.e., Z_2) have $r = 1$ distinct value. Thus, by Theorem 6.2, the number of distinct column functions is at most

$$1+(2+1)=4.$$

■

Theorem 6.3: Let f be an n-variable positive threshold logic function with r distinct non-zero weights. Let r be a constant that is independent of n. Then, the number of 6-LUTs to realize f is $O(n^{r+1}\log n)$.

(Proof) To implement f, use cells with $K = R+1$ inputs, where $R = \lceil \log_2 \mu(f) \rceil$. To realize f by an LUT cascade with K-input cells, we need $s = n - R$ cells, by Lemma 2.2. Note that each cell has at most R outputs. The number of 6-LUTs to synthesize a K-input LUT is $M \simeq 2^{K-4}/3$, by Theorem 2.3. Note that $M \simeq \frac{2^R}{24}$. Thus, the total number of 6-input LUTs to realize f is

$$N_{LUT} \le sRM \le n\lceil \log_2 \mu(f) \rceil \frac{\mu(f)}{24}.$$

Note that $\mu(f)$ is $O(n^r)$, by Theorem 6.2. Hence, we have the theorem. □

Thus, for binary weight neural networks ($r = 1$), all the non-zero weights are 1, and the number of 6-LUTs is $O(n^2 \log n)$.

Theorem 6.4: In an n-variable threshold logic function with $(1, 2)$ weights, the column multiplicity of decomposition chart can be reduced to at most $n+1$ by reordering of the variables.

(Proof) For simplicity, assume that $n = 2m$. Let the input variables be $X = (x_1, x_2, \ldots, x_n)$, and let (X_1, X_2) be a partition of X. Let the weights of X_1 and X_2, be 1 and 2, respectively. Let n_1 be the number of variables in X_1, and let n_2 be the number of variables in X_2. Let k be the number of bound variables.

When $k \le n_1$. The weights of the bound variables are the same. Thus, the column multiplicity is at most $k+1$. Especially, when $k \le m$, the theorem holds. On the other

		0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	x_4
		0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	x_3
		0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	x_2
		0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	x_1
0	0																1	
0	1						1	1	1		1	1	1	1	1	1	1	
1	0						1	1	1		1	1	1	1	1	1	1	
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
x_6	x_5																	

Fig. 6.2. Decomposition chart for $f(Z_1, Z_2)$, where $Z_1 = (x_1, x_2, x_3, x_4)$ and $Z_2 = (x_5, x_6)$

hand, when $m < k \leq n_1$, the number of free variables is at most $m-1$, and free variables have different weights. By Theorem 3.2, the column multiplicity is at most $n+1$.

When $k > n_1$. The bound variables have different weights, but the weights of the free variables are the same. Thus, the number of distinct column functions is at most $n_2 + 2$. Thus, when $n_2 \leq m$, the theorem holds. On the other hand, when $n_2 > m$, the number of bound variables is at most $m-1$, but variables with different weights exist. By Theorem 3.1, the column multiplicity is at most n. □

Corollary 6.1: Any n-variable threshold logic function with $(1, 2)$ weights, can be realized with $O(n^2 \log n)$ 6-LUTs.

VII. Application to Ternary Weight Neural Networks

In neural networks, weights of threshold gates are real numbers. To implement them by digital circuits, weights are quantized. When the weight is small, it is quantized to 0. Thus, threshold gates often have zero weights.

In neural networks using $(-1, 0, 1)$ weights, more than 90% of the weights are 0 [22]. This is quite desirable, since the function is independent of the variables whose weights are 0, and no connection is necessary for such variables. Such a circuit is called **ternary weight neural network** [10]. From this property, the circuit can be reduced drastically.

When $(-1, 0, 1)$ weights are used, a threshold gate realizes a function g that is 1 if and only if

$$-(y_1 + y_2 + \ldots + y_k) + (y_{k+1} + y_{k+2} + \ldots + y_m) \geq T_1,$$

where the weights for $\{y_1, y_2, \ldots, y_k\}$ are -1, while the weights for $\{y_{k+1}, y_{k+2}, \ldots, y_m\}$ are 1.

However, by replacing y_i with $-z_i$ for $i \in \{1, 2, \ldots, k\}$, we have another threshold logic function h that is 1 if and only if

$$(z_1 + z_2 + \ldots + z_k) + (y_{k+1} + y_{k+2} + \ldots + y_m) \geq T_2.$$

Note that the function h is totally symmetric.

From Theorem 5.2 and Theorem 6.3, we have the following:

Theorem 7.1: Any n-variable threshold logic function f with $(-1, 0, 1)$ weights, can be realized with $O(n^2 \log n)$ 6-LUTs. Especially, f can be realized using the following number of 6-LUTs:

- $n + 1$ (when $n \leq 21$).
- $2n - 19$ (when $22 \leq n \leq 31$)
- $5n - 113$ (when $32 \leq n \leq 62$).

Example 7.1: Let $f(X_1, X_2)$ be the 18-variable threshold function, where $X_1 = (x_1, x_2, \ldots, x_9)$, $X_2 = (x_{10}, x_{11}, \ldots, x_{18})$, and the weights of the variables in X_1 and X_2 be -1 and 1, respectively. Then, f can be realized by the cascade shown in Fig. 6.1. When the threshold of the function f is $T = 0$, the profile of the function is

$$(2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 9, 8, 7, 6, 5, 4, 3, 2).$$

The number of 6-LUTs to realize f is 17. ■

From Corollary 6.1, we have the following:

Theorem 7.2: Any n-variable threshold logic function with $(-2, -1, 0, 1, 2)$ weights, can be realized with $O(n^2 \log n)$ 6-LUTs.

VIII. Related Works

A. Threshold Logic Functions

[11] showed the list of optimum structures for representative threshold logic functions of up to 6 variables.

[21] and [13] enumerated threshold logic functions of up to 7 and 8 variables, respectively. They used linear integer programming to obtain optimum structures of threshold logic functions. When $n \leq 7$, for any function, the optimal structure is unique, and weights are integers. However, when $n = 8$, there exist non-integer weights, and some functions have multiple optimal solutions [14].

[9] considered an efficient method to enumerate the number of positive functions, which is a superset of positive threshold logic functions.

[12] showed a series of n-variable threshold logic functions whose weights increase exponentially with n. However, a function with large weights does not always have a large C-measure. For example, the C-measure of the 16-variable threshold logic function with structure $(30, 32, 59, 62, 117, 121, 233, 238, 463, 471, 925, 934, 1844, 1859, 3686, 3703; 7389)$ is only 11.

[7] showed a series of n-variable threshold logic functions whose C-measures increase exponentially with n, even if the ordering of the variables is optimized. For example, the C-measure of the 16-variable threshold logic function with the structure $(17, 18, 20, 24, 33, 34, 36, 40, 65, 66, 68, 72, 129, 130, 132, 136; 510)$ is 55.

[1] showed a method to decompose a threshold logic function into bounded fan-in threshold logic functions. The

method produces circuits whose sizes are polynomial of n. However, the circuit structure depends on the value of the threshold T.

B. Neural Networks

[5] developed a binary neural network, where both activations and weights are binary. [8] and [10] developed ternary weight neural networks, where activations are binary. Ternary weight neural networks yield higher accuracy (i.e., have fewer errors for unknown data) than binary ones. [23] and [6] introduced methods to train ternary neural net. [19] developed ternary weight neural networks and implemented them on a table look-up based processor, where the ternary weights are $\{-1, 0, 1\}$. Note that, when the weight is 0, no connection is necessary. In some applications, more than 70% of the weights are zero.

IX. Conclusions and Comments

In this paper, we improved upper bounds on the column multiplicity of decomposition chart for a threshold logic function. With these results, we showed an efficient method to realize an n-variable threshold logic function by an LUT cascade. In a conventional method, the size increases exponentially with n, while in the presented method, the size increases polynomially: When the number of distinct non-zero weights is r, the size increases with $O(n^{r+1} \log n)$.

Especially, n-variable threshold logic functions with weights $(-1, 0, 1)$ or $(-2, -1, 0, 1, 2)$ can be realized with $O(n^2 \log n)$ K-LUTs. By only changing the content of LUTs, we can realize different threshold functions sharing the same set of weights $(w_1, w_2, \ldots, w_n)$, but having the different threshold T. With these results, we can estimate the size of an FPGA necessary to implement a given ternary weight neural network. Note that LUT cascades are easy to layout.

When $n \leq 60$, cascade realizations are efficient. However, when n is large, tree-type realizations can be more efficient [1].

Acknowledgments

This work was supported in part by a Grant-in-Aid for Scientific Research of the JSPS. The author thanks Prof. H. Nakahara for discussion. Reviewer's comments were also useful for the improvement of the paper.

References

[1] V. Annampedu and M. D. Wagh, "Decomposition of threshold functions into bounded fan-in threshold functions," *Information and Computation*, 227, June 2013, pp. 84-101.

[2] R. L. Ashenhurst, "The decomposition of switching functions," *International Symposium on the Theory of Switching*, pp. 74-116, April 1957.

[3] M. Blott, T. B Preuser, N. J. Fraser, G. Gambardella, K. O. Brien, Y. Umuroglu, M. Leeser, and K. Vissers, "FINN-R: An end-to-end deep-learning framework for fast exploration of quantized neural networks," *ACM Transactions on Reconfigurable Technology and Systems*, Vol. 11, No. 3, Dec. 2018, pp.1-23.

[4] H. A. Curtis, *A New Approach to the Design of Switching Circuits*, D. Van Nostrand Co., Princeton, NJ, 1962.

[5] M. Courbariaux, Y. Bengio, and J.P. David, "BinaryConnect: Training deep neural networks with binary weights during propagations," *Advances in Neural Information Processing Systems*, pp. 3123-3131, 2015.

[6] Z. He and D. Fan, "Simultaneously optimizing weight and quantizer of ternary neural network using truncated Gaussian approximation," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (CVPR-2019), Long Beach, CA, USA, 2019 pp. 11430-11438.

[7] K. Hosaka, Y. Takenaga, and S. Yajima, "On the size of ordered binary decision diagrams representing threshold functions," *Theoretical Computer Science*, Vol. 180, Issues 1-2, 10 June 1997, pp. 47-60.

[8] K. Hwang and W. Sung, "Fixed-point feedforward deep neural network design using weights +1, 0, and +1," *2014 IEEE Workshop on Signal Processing Systems*,(SiPS), Oct. 2014, pp. 1-6.

[9] D. Kleitman, "On Dedekind's problem: The number of monotone Boolean functions,' '*Proc. Amer. Math. Soc.*, Vol. 21, pp. 677-682, 1969.

[10] F. Li, B. Zhang, and B. Liu, "Ternary weight networks," *30th Conference on Neural Information Processing Systems* (NIPS 2016), Barcelona, Spain 2016.

[11] S. Muroga, I. Toda and M. Kondo, "Majority decision functions of up to six variables," *Mathematics of Computation*, Vol. 16, No. 80, Oct. 1962, pp. 459-472.

[12] S. Muroga, "Lower bounds of the number of threshold functions and a maximum weight," *IEEE Transactions on Electronic Computers*,Vol. EC-14, Issue: 2, April 1965, pp.136-148.

[13] S. Muroga, T. Tsuboi, and C. Baugh, "Enumeration of threshold functions of eight variables," *IEEE Transactions on Computers*, Vol.C-19, No.9, Sept. 1970, pp.818-825.

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

[15] T. Sasao, "Analysis and synthesis of weighted-sum functions," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 25, No. 5, May 2006, pp. 789 - 796.

[16] T. Sasao, *Switching Theory for Logic Synthesis*, Kluwer Academic Publishers, 1999.

[17] T. Sasao, *Memory-Based Logic Synthesis*, Springer, 2011.

[18] T. Stephen and T. Yusun, "Counting inequivalent monotone Boolean functions,"*Discrete Applied Mathematics*, Vol. 167, April 2014, pp.15-24.

[19] Y. Suzuki, N. Soga, S. Sato, and H. Nakahara, "A table look-up based ternary neural network processor," *2020 IEEE 50th International Symposium on Multiple-Valued Logic*, (ISMVL-2020), Nov. 2020, pp. 188-193.

[20] Y. Umuroglu, Y. Akhauri, N. J. Fraser, M. Blott, "LogicNets: Co-designed neural networks and circuits for extreme-throughput applications,"*30th International Conference on Field-Programmable Logic and Applications*, FPL 2020, pp. 291-297.

[21] R. Winder,"Enumeration of seven-argument threshold functions," *IEEE Trans. Electron. Comput.*, Vol. 14, 1965, pp. 315-325.

[22] H. Yonekawa, S. Sato, and H. Nakahara, "A ternary weight binary input convolutional neural network: Realization on the embedded processor," *2018 IEEE 48th International Symposium on Multiple-Valued Logic*, (ISMVL-2018), May 16-18. 2018, pp. 174-179.

[23] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization,"*5th International Conference on Learning Representations*, ICLR (Poster), 2017.