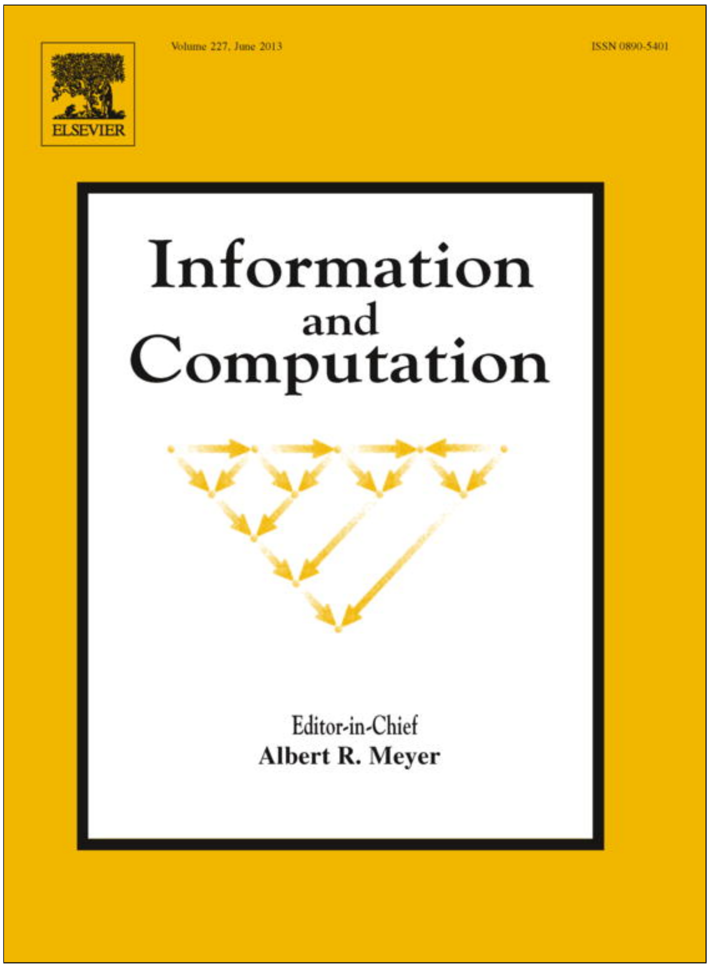
Volume 227, June 2013
ISSN 0890-5401
ELSEVIER
Information and Computation
Editor-in-Chief
Albert R. Meyer

Information and Computation 227 (2013) 84–101

Contents lists available at SciVerse ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

Decomposition of threshold functions into bounded fan-in threshold functions [☆]

Viswanath Annampedu[a], Meghanad D. Wagh[b,*]

[a] Serdes Architecture, LSI Corp., Allentown, PA 18109, United States
[b] Department of Electrical and Computer Engineering, Lehigh University, Bethelehem, PA 18015, United States

ARTICLE INFO

Article history:
Received 31 October 2007
Revised 3 March 2011
Available online 10 April 2013

Keywords:
Threshold functions
Decomposition
Bounded fan-in
Majority logic
Comparison function

ABSTRACT

This paper obtains explicit decomposition of threshold functions into bounded fan-in threshold functions. A small fan-in is important to satisfy technology constraints for large scale integration. By employing the concept of error in the threshold function, we are able to decompose functions in $\widehat{LT}_1$ into a network of size $O(n^c/M^2)$ and depth $O(\log^2 n/\log M)$ where n is the number of inputs of the function and M is the fan-in bound. The proposed construction enables one to trade-off the size and depth of the decomposition with the fan-in bound. Combined with the work on small weight threshold functions, this implies polynomial size, $\log^2$ depth bounded fan-in decompositions for arbitrary threshold functions in LT_d. These results compare favorably with the classical decomposition which has a size $O(2^{n-M})$ and depth $O(n-M)$.

We also show that the decomposition size and depth can be significantly reduced by exploiting the relationships between the input weights. As examples of this strategy, we demonstrate an $O(n^2/M)$ size decomposition of the majority function and, $O(n/M)$ size decompositions of an error tolerant pattern matching function and the comparison function. In all these examples, except for the first level, all other levels use only majority functions.

1. Introduction

1.1. Background

A Boolean function $\{0,1\}^n \to \{0,1\} : f(x_1, x_2, \dots, x_n)$ is called *threshold* if there exist real numbers $w_1, w_2, \dots, w_n$ and T such that

$$f(x_1, x_2, \dots, x_n) = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} w_i x_i \geqslant T, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

The constants $w_1, w_2, \dots, w_n$ are called the *weights* of the inputs and T is called the *threshold*. We denote a threshold function as $\mathbf{TH}(x_1, x_2, \dots, x_n;\ w_1, w_2, \dots, w_n;\ T)$. Muroga has shown that any threshold function can be realized using integer weights and threshold [1]. Therefore our analysis in this paper will only use integer values for these quantities. A threshold function of n variables with unit weights and a threshold equal to $\lfloor n/2 \rfloor + 1$ is known as a *majority function*. A threshold function with unit weights (but any threshold) is called a generalized majority function. Reader is cautioned that

[☆] This work was supported in part by NSF under Grant ECCS-0925890.
[*] Corresponding author.
E-mail addresses: Vish.Annampedu@lsi.com (V. Annampedu), mdw0@lehigh.edu (M.D. Wagh).

0890-5401/$ – see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ic.2013.04.002

in some literature a threshold function with weights ±1 is referred to as a generalized majority function and sometimes, even as a majority function. All the results of this paper use the stricter definitions presented here.

As is common with literature in threshold functions, this paper uses equations that mix arithmetic and Boolean variables (see (1)). The confusion between arithmetic multiplication/addition and identical Boolean AND/OR symbols can generally be resolved from the context. In particular, when one or more of the operands in an equation is a real number (e.g., the weights or the threshold), then all operations in that equation are considered arithmetic while if all the operands are Boolean, then all operations are considered Boolean.

Realizations of threshold functions are called threshold gates. A variety of physical effects can be employed to create a weighed sum of the inputs and then compare it with a preset threshold. These include voltage or current scaling and summation using operational amplifiers [1], charge deposition and summation using a parallel capacitance network [2] and more recently, direct implementation of threshold logic using nanoelectronic devices such as resonant tunneling diodes/transistors (RTD/RTT) [3], single electron transistors (SET) [4] and quantum-dot cellular automata (QCA) [5].

Threshold functions are often classified based on their implementations as follows. The class of Boolean functions that can be implemented by single threshold gates with unbounded fan-in[1] is known as LT_1. In general, Boolean functions computable by depth d networks of unbounded fan-in threshold gates belong to class LT_d. The size (number of threshold gates) of an LT_d implementation is restricted to a polynomial function of the number of inputs. Members of the class LT_d which have weights of polynomial order form a subclass denoted by $\widehat{LT}_d$. Clearly functions in $\widehat{LT}_d$ are more realistic from the implementation standpoint. The class of Boolean functions which can be realized by constant depth, polynomial size networks of threshold functions with ±1 weights is denoted by TC^0. Since any function in $\widehat{LT}_1$ can be converted to a threshold function with ±1 weights by duplicating inputs, it follows that $\widehat{LT}_d \subseteq TC^0$ for any constant d [6]. The only class that uses threshold gates with bounded[2] fan-in is the class NC^k. This class consists of Boolean functions that can be implemented as a polynomial size, depth $O((\log n)^k)$ network of bounded fan-in AND, OR and NOT gates. It is known that $TC^0 \subseteq NC^1$ [7,8]. Thus any function in $\widehat{LT}_1$ can be implemented using a $\log n$ depth network of three specific types of bounded fan-in threshold gates; AND, OR and NOT. However, it is long known that threshold functions are very powerful and a single threshold gate can often replace a complex network of AND, OR and NOT gates. To exploit the power of threshold functions, this paper provides explicit decomposition of any member of LT_d into a network of arbitrary threshold gates with bounded fan-in.

A large number of arithmetic circuits including adders, multipliers, dividers, iterated adders and multipliers, powering circuits and signed digit operations have been realized using networks of threshold gates [9–11]. In addition, many applications such as counters, comparators and error tolerant pattern matching have been implemented through single threshold functions [12–14]. Neural networks can also be modeled using threshold gates. Further, recent developments in nanotechnology suggest that threshold functions may be the foundation for realizing Boolean functions of the future [3,15]. The results of this paper are applicable to all these applications and are consistent with the current and future technology constraints.

When a function is implemented using a network of threshold gates, certain network parameters are crucial in determining its performance and practicality. The depth of the network directly controls the delay of the final realization. The total number of threshold gates used in the network determines the cost of the realization. The weights and the threshold influence the area, power and reliability of the threshold gate. Finally, the fan-in, i.e., the number of inputs to a threshold gate, determines its suitability for implementation and limits its reliability. It is therefore imperative that all these network parameters be minimized so as to derive maximum benefit out of these networks.

Since network delay is perceived as the most critical parameter, much of the research in this area is focused on developing small depth threshold networks for various applications [16–18]. Several important results are also available to implement any Boolean function in a three level network of generalized majority functions [19,20]. Finally, it has also been shown that any LT_d member function can be decomposed into an $\widehat{LT}_{d+1}$ function [6,21]. In all these studies, the network size is used as a quality measure.

Converting a network of threshold functions into one that employs threshold functions with bounded fan-in has received relatively less attention [7]. Generally, the constant depth implementations result in decompositions with fan-ins that are dependent upon the number of inputs [8]. Unfortunately, threshold gate fan-in is a bigger constraint than the network depth in the current VLSI and the future nanotechnology implementations of threshold networks [22]. For example, experimental studies suggest that the reliability considerations of threshold gates based on resonant tunneling devices may force one to restrict the fan-in to as small a value as 5 or 6 [23,24]. On the other hand, networks based on such gates can be easily pipelined so that the network throughput is independent of the depth of the network [23].

This paper discusses decomposition of a threshold function into a network of threshold functions, each with a bounded fan-in. This decomposition employs the novel concept of *error* of the threshold function. We show that the value of the error is always non-negative and can be obtained by adding (non-negative) errors due to independent groups of inputs. When the total error of a threshold function exceeds the *critical error*, the output of the function is 0, otherwise it is 1. Critical error is dependent on the weights and the threshold of the function, and plays a central role in the design of our network. The

[1] Fan-in of a function or a gate refers to the number of inputs.

[2] In this paper, a bounded fan-in always refers to a fan-in bound which is independent of the number of inputs.

network can be visualized as made up of *fragments* which compute the errors of different groups of inputs and a network of *recombiners* which add these errors to eventually compare it to the critical error.

1.2. Contributions of this paper

This paper gives a constructive proof of decomposing any function in $\widehat{LT}_1$ into bounded fan-in threshold gates on the first level (Theorem 1) and bounded fan-in generalized majority gates on the rest (Theorem 4). The decomposition has a polynomial size and a $\log^2$ depth in terms of the number of inputs.

The decomposition technique developed here allows one to exploit the relationships between weights of inputs to substantially reduce the size and depth of the decomposition (Theorems 5–10). Applying these results, we show that: 1. An n-input majority function can be decomposed into a network of bounded fan-in threshold functions with size $O(n^2/M)$ and depth $O(\log^2 n/\log M)$ where M is the bound on the fan-in (Theorem 11); 2. A network of bounded fan-in threshold functions of size $O(n/M)$ and depth $O(\log(n/M))$ can realize the n-bit error tolerant pattern matching function for small error tolerances (Theorem 12); and 3. An n-bit binary number comparison function can be decomposed into a network of bounded fan-in threshold functions with size $O(n/M)$ and depth $O(\log(n/M))$ (Theorem 13).

Using the decomposition methods developed here, one can trade-off the size and depth of the decomposition with the bound on the fan-in. In particular, we show that the size decreases as the square of the fan-in and the depth, as the logarithm of the fan-in.

As an aside, this paper also shows that any threshold function in $\widehat{LT}_1$ can be implemented as a depth 2 network which has bounded fan-in threshold functions (with any chosen fan-in) at the first level and a generalized majority function at the second level (Theorem 2).

1.3. Organization of the paper

The overview of our general decomposition strategy is presented in Section 2. It shows that each fragment is a collection of bounded fan-in threshold functions. We also show that an arbitrary threshold function has a two level decomposition where the first level is made up of bounded fan-in threshold functions and the second level is a generalized majority function. Section 3 shows that a recombiner is composed of a collection of majority functions, each of which may be realized through its own network of bounded fan-in threshold functions. The size of our threshold network is generally dependent on the critical error of the function being decomposed. However, by appropriate grouping of the inputs based upon their weights, one can often reduce this size significantly. Section 4 provides several theorems which are useful in reducing the network size based on intelligent grouping of inputs. Section 5 presents three examples to demonstrate the decomposition and the complexity reduction strategies. Finally the conclusions of this work are provided in Section 6.

2. Decomposition strategy

The classical decomposition of threshold functions exploits the fact that threshold functions are unate.[3] A unate function $f(x_1, x_2, \ldots, x_n)$ can be decomposed as [25]

$$f(x_1, x_2, \ldots, x_n) = \begin{cases} f_1(x_2, x_3, \ldots, x_n) + x_1 f_2(x_2, x_3, \ldots, x_n) & \text{if } x_1 \text{ is positive,} \\ f_1(x_2, x_3, \ldots, x_n) + \bar{x}_1 f_2(x_2, x_3, \ldots, x_n) & \text{if } x_1 \text{ is negative.} \end{cases} \tag{2}$$

If f in (2) is threshold, then so are the $n-1$ variable functions f_1 and f_2. Function f can therefore be decomposed into three threshold functions, f_1, f_2 and a 3-input threshold function that combines f_1, f_2 and x_1. Eq. (2) can be employed repeatedly to reduce the fan-in of the threshold functions to any desired small value M $(< n)$. The final decomposition results into a binary tree of $n-M$ levels with $2^{n-M}-1$ internal nodes, each representing the 3-input threshold function and 2^{n-M} leaves representing M-input threshold functions (not necessarily distinct). Clearly, this decomposition has a depth of $(n-M+1)$ which is $O(n-M)$ and a size of $O(2^{n-M})$. In addition to the large size, the classical decomposition also suffers from the fact that the M inputs x_{n-M+1} to x_n that are not used in the decomposition are potentially used in the 2^{n-M} threshold functions at the tree leaves. Each of the other input variable x_i, $1 \leqslant i \leqslant n-M$, is used in 2^{i-1} threshold functions. Driving a large number of gates implies excessive load on inputs in practical applications.

In this paper we propose an alternate decomposition of threshold functions into a network of bounded fan-in threshold functions. Our decomposition has a polynomial size (with respect to n) and each input drives a small number of gates. Our scheme, shown in Fig. 1, consists of partitioning the inputs into disjoint sets, each of which is processed through an independent multi-output logic block at the top level. All the outputs of each pair of blocks on any level are combined by multi-output logic blocks on the next level. We call the top level blocks to which inputs are directly applied as the *Fragments* and the blocks on every other level as the *Recombiners*. We will show later that each output of a fragment is a threshold function of the inputs to that fragment (Theorem 1). Similarly, each output of a recombiner is a majority

[3] A *unate* function is a Boolean function in which every variable is either positive or negative. A variable (say) x_1 is positive in $f(x_1, x_2, \ldots, x_n)$ if $f(1, x_2, \ldots, x_n) \geqslant f(0, x_2, \ldots, x_n)$ for all $x_2, x_3, \ldots, x_n$. It is negative, if $f(1, x_2, \ldots, x_n) \leqslant f(0, x_2, \ldots, x_n)$.

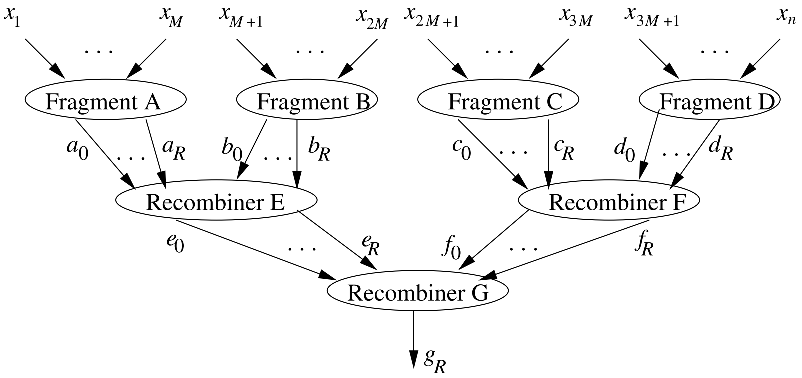

Fig. 1. The decomposition of a threshold function $f(x_1, x_2, \dots, x_n)$ with weights $w_1, w_2, \dots, w_n$ and threshold T in multi-output logic blocks. Each of the $R+1$ outputs of a block are threshold functions of the inputs to the block. $R = (\sum_{w_i>0} w_i) - T$.

function of a subset of inputs to that recombiner (Theorem 3). A selected output of the recombiner in the last level gives the value of the threshold function being decomposed. The output of a threshold function is determined by the comparison of a weighed sum of its inputs, $\sum w_i x_i$, with the threshold T. One simple way to realize this computation is to let each fragment calculate the weighed sum of its inputs. The recombiners can then add these sums and compare the total with the threshold to determine the output value. Unfortunately, the weighed sum of the inputs to a fragment can be as small as the sum of the negative weights and as large as the sum of the positive weights. Since these sums lie in different ranges, transmitting and combining them is complicated.

In this paper, we take a different approach. Instead of using the weighed sums of the inputs, we use the *error* of the threshold function to determine its output. Error E of a threshold function $f(x_1, x_2, \dots, x_n)$ is defined as

$$E = K - \sum_{i=1}^{n} w_i x_i, \quad \text{where } K = \sum_{\substack{i=1 \\ w_i>0}}^{n} w_i. \tag{3}$$

Clearly, $E \geqslant 0$ because K is the largest value of $\sum w_i x_i$. Let constant $R = K - T$, where T is the threshold of the function being decomposed. It is easy to see that $R \geqslant 0$; otherwise the function would always be 0. Combining (1) and (3) one gets

$$f(x_1, x_2, \dots, x_n) = \begin{cases} 1 & \text{if } E \leqslant R, \\ 0 & \text{if } E > R. \end{cases} \tag{4}$$

Eq. (4) gives an alternate way to compute the output of a threshold function. We call R, the *critical error* of the threshold function.

In the proposed decomposition, we compute the error of fragment j with inputs x_i, $i \in S_j$, as

$$E_j = K_j - \sum_{i \in S_j} w_i x_i, \quad \text{where } K_j = \sum_{\substack{i \in S_j \\ w_i>0}} w_i. \tag{5}$$

Note that similar to E, E_j is also non-negative. By adding E_js of all the fragments one gets

$$\sum_j E_j = \sum_j K_j - \sum_j \sum_{i \in S_j} w_i x_i = K - \sum_{i=1}^{n} w_i x_i = E. \tag{6}$$

Our network is designed to add the E_js to obtain E. Further, since one is only interested in comparing E to R, we truncate the value of error at every stage of calculation at $R+1$. Thus the error values within our network lie between 0 and $R+1$ (inclusive of both) and are transmitted using $R+1$ Boolean variables.

Boolean output p_t, $0 \leqslant t \leqslant R$, of a fragment is defined as

$$p_t = \begin{cases} 1 & \text{if error } E_j \text{ in that fragment} \leqslant t, \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

Fragment j specifies error E_j using its $R+1$ outputs, p_i, $0 \leqslant i \leqslant R$. An error value e is indicated by setting outputs $p_0, p_1, \dots, p_{e-1}$ to 0 and $p_e, p_{e+1}, \dots, p_R$ to 1. Note that $e > R$ is indicated by $p_i = 0$, for all i, $0 \leqslant i \leqslant R$. Using $(R+1)$ output variables to carry $\log_2(R+2)$ bits of information seems wasteful, but it allows each of these variables to be threshold functions of inputs as shown in Theorem 1.

The p_ts defined by (7) are related as

$$p_i \leqslant p_j \quad \text{if } i < j. \tag{8}$$

Eq. (8) has the following implications which are used later:

$$p_i \cdot p_j = p_i \quad \text{and} \quad p_i + p_j = p_j \quad \text{if } i < j. \tag{9}$$

We conclude this section with the following important observation [26] that follows directly from (5) and (7).

Theorem 1 *(Fragment outputs). Each output p_t of fragment j is a threshold function of inputs x_i with weights w_i, $i \in S_j$, and a threshold of $K_j - t$.*

The outputs of all the fragments as defined here may be combined by a single threshold gate to provide a two level decomposition of any threshold function as given in the following theorem.

Theorem 2 *(Two level decomposition). Any threshold function f of n variables may be decomposed into a depth 2 threshold network such that the first level has threshold gates with fan-in bounded by any M and the second level has a generalized majority gate with fan-in of $(R+1)\lceil n/M \rceil$, where R denotes the critical error of f.*

Proof. Function f may be decomposed to use fragments with fan-in M on the first level. Let the output i of the fragment j be denoted by $f(j,i)$, $1 \leqslant j \leqslant (n/M)$, $0 \leqslant i \leqslant R$. We now show that f is obtained by combining all $f(j,i)$ in a generalized majority gate with a threshold of $\lceil n/M \rceil(R+1) - R$. From the definition (7) of the fragment outputs, one can see that the number of variables in the set $\{f(j,i) \mid 0 \leqslant i \leqslant R\}$ that are equal to 1 is $R + 1 - E_j$ where E_j is the error in the jth fragment. (This number is 0 if $R + 1 - E_j$ is negative.) Thus the total number of nonzero outputs from all the fragments equals (at most) $\lceil n/M \rceil(R+1) - E$, where E is the total error in the function. But from (4), $f = 1$ only when $E \leqslant R$. Thus $f = 1$ only when at least $\lceil n/M \rceil(R+1) - R$ of the outputs from all the fragments are 1. Thus the function on the second level of decomposition is a generalized majority gate with threshold $\lceil n/M \rceil(R+1) - R$. □

The size of the decomposition described in Theorem 2 is $1 + \lceil n/M \rceil(R+1)$ threshold gates. If the function being decomposed is in $\widehat{LT}_1$, then R is polynomial in n and therefore so is the size of the decomposition. Yao has previously proved that functions in ACC^0 class[4] can be implemented using a two level network of size $2^{(\log n)^c}$ for some constant c [7,27]. The first level of this decomposition has AND gates of fan-in $O((\log n)^c)$ and the second level has a symmetric function gate (which can be implemented by a two level generalized majority gate network).

3. Recombiner implementation

Neither the symmetric function gate in Yao's implementation [27] nor the generalized majority gate in the two level decomposition of Theorem 2 has a bounded fan-in. To achieve the desired fan-in, we use a multistage network using the recombiner blocks to add E_js as shown in Fig. 1.

Recall that each recombiner is a multi-output logic block and uses as its inputs, all the outputs from two blocks (its *parents*) on the previous level. The $R + 1$ outputs of a recombiner denote an error value in a manner similar to the outputs of a fragment. The error value they indicate equals the sum (truncated at $R+1$) of the error values provided by the parents. This section shows that each output of a recombiner is a majority function of a subset of its inputs and can be easily decomposed into threshold functions with bounded fan-in.

Let p_i, q_i, $0 \leqslant i \leqslant R$ represent the inputs to a recombiner from its two parents and s_t, $0 \leqslant t \leqslant R$ its outputs. Similar to the fragment output definition (7), we define output s_t to be 1 if and only if the sum of the two error values provided by the parents is t or less. One can then see that the Boolean expression for s_t is

$$s_t = \sum_{i=0}^{t} p_i q_{t-i}, \quad 0 \leqslant t \leqslant R. \tag{10}$$

Relation (10) may be justified by noting that the term $p_i q_{t-i}$ is (logical) 1 if and only if one parent indicates an error $\leqslant i$ and the other, an error $\leqslant t - i$. Thus, each term of the summation (10) accounts for a case when the sum of errors indicated by parents is $\leqslant t$. Each term of (10) is (logical) 0 if the combined error from the two parents is greater than t. As an example, consider a case when $R = 4$, $[p_0, p_1, p_2, p_3, p_4] = [0, 1, 1, 1, 1]$ and $[q_0, q_1, q_2, q_3, q_4] = [0, 0, 1, 1, 1]$. Note that these parents are indicating 1 and 2 errors respectively. Eq. (10) then gives $[s_0, s_1, s_2, s_3, s_4] = [0, 0, 0, 1, 1]$ showing that the total error indicated by the two parents of the recombiner is 3.

Note that because of the way the error values are combined (see Fig. 1), the output of any recombiner provides the total error indicated by *all* the fragments of whom it is a descendant.

Theorem 3 describes the threshold nature of each output of a recombiner.

[4] Class ACC^0 refers to functions realizable as a polynomial size, constant depth network of unbounded fan-in AND, OR, NOT and a finite set of MOD$_m$ gates [7].

Theorem 3 *(Recombiner outputs). Function s_t defined by (10) is a majority function.*

Proof. Since arguments p_i and q_i of s_t satisfy condition (8), there exist integers a and b such that

$$p_i = \begin{cases} 0 & \text{if } 0 \leqslant i < a, \\ 1 & a \leqslant i \leqslant t, \end{cases} \tag{11}$$

and

$$q_i = \begin{cases} 0 & \text{if } 0 \leqslant i < b, \\ 1 & b \leqslant i \leqslant t. \end{cases} \tag{12}$$

Therefore, from the expression (10) of s_t,

$$s_t = \sum_{i=0}^{t} p_i q_{t-i} = \sum_{i=a}^{t} q_{t-i} = q_{t-a}.$$

The last step of the above equation follows directly from (9). Thus $s_t = 1$ if and only if $t - a \geqslant b$, i.e., if $(t+1-a) + (t+1-b) \geqslant t+2$. However, from (11) one can see that $(t+1-a)$ is simply the number of p_is in the summand of s_t that are 1. Similarly, from (12) $(t+1-b)$ represents the number of q_is in the summand of s_t that are 1. Therefore $s_t = 1$ when at least $t+2$ of its inputs (out of a total of $2(t+1)$ inputs) are equal to 1 thus showing that s_t is a majority function. □

Theorem 3 shows that each recombiner output s_t, $0 \leqslant t \leqslant R$, given by (10) is a threshold (in fact, a majority) function. However, its fan-in may exceed the fan-in bound. Following corollary helps decompose s_t into bounded fan-in threshold functions.

Corollary 1. *Let Boolean variables x_i, y_i, $0 \leqslant i \leqslant m-1$ be such that*

$$x_0 \leqslant x_1 \leqslant \cdots \leqslant x_{m-1} \quad \textit{and} \quad y_0 \leqslant y_1 \leqslant \cdots \leqslant y_{m-1}.$$

Then

(a) $\sum_{i=1}^{m-1} x_i y_{m-i}$ *is a majority function of* $2m-2$ *inputs.*
(b) $\sum_{i=1}^{m-1} x_i y_{m-i} + y_0$ *is a majority function of* $2m-1$ *inputs.*
(c) $\sum_{i=1}^{m-1} x_i y_{m-i} + x_0 + y_0$ *is a threshold function of* $2m$ *inputs with unit weights for all inputs and a threshold of* m.

Proof. Part (a) of the corollary can be proved similar to Theorem 3.

To prove part (b), consider a majority function with $2m-1$ inputs x_1 through x_{m-1} and y_0 through y_{m-1} with all weights equal to 1 and threshold equal to m. We will prove that this majority function always gives the same output as the given Boolean function. If input y_0 is 0, then from part (a), the two functions are the same. If $y_0 = 1$, the Boolean function is 1. But when $y_0 = 1$, so are y_1 through y_{m-1}. Since m inputs of the majority function are 1, its output is also equal to 1. Thus the Boolean function $\sum_{i=1}^{m-1} x_i y_{m-i} + y_0$ is identical to the majority function constructed.

To prove part (c) in a similar fashion, consider a threshold function with unit weights for all $2m$ inputs x_0 through x_{m-1} and y_0 through y_{m-1} and a threshold equal to m. We will prove that it has identical outputs as the Boolean function $\sum_{i=1}^{m-1} x_i y_{m-i} + x_0 + y_0$. Note that when $x_0 = 0$, the equality of the two functions is established by part (b). When $x_0 = 1$, the Boolean function produces a 1. But $x_0 = 1$ also implies that x_1 through x_{m-1} are all 1 and consequently the threshold function is also 1. Thus the threshold function is the same as the Boolean function. □

To decompose a recombiner output s_t into bounded fan-in threshold functions, note that s_t is a function of p_i, q_i, $0 \leqslant i \leqslant t$ as given in (10). Relation (8) shows that the Boolean variables p_i and q_i satisfy the conditions on x_is and y_i in the Corollary 1. Thus this corollary can be applied to any subset of p_is and q_is. In particular, when the fan-in bound M is even, Corollary 1(a) can be applied with $m = M/2 + 1$ to variables involved in (10) for every consecutive m values of i. As an example, consider the decomposition of s_7 when $M = 4$. Let $m = 3$, $x_1 = p_0$, $x_2 = p_1$ and $y_1 = q_6$, $y_2 = q_7$. Then Corollary 1(a) shows that $p_0 q_7 + p_1 q_6$ is a majority function. Similarly using $x_1 = p_2$, $x_2 = p_3$ and $y_1 = q_4$, $y_2 = q_5$ shows that $p_2 q_5 + p_3 q_4$ is a majority function. Proceeding in this manner, one also can prove that $p_4 q_3 + p_5 q_2$ and $p_6 q_1 + p_7 q_0$ are also a majority functions. Function s_7 given in (10) can then be obtained by ORing these four majority functions as shown in Fig. 2. Since each gate in this structure combines M operands into a single operand, one needs $\lceil (2t+1)/(M-1) \rceil$ gates to convert $(2t+2)$ inputs of s_t into a single output. It is also easy to see that the depth of the decomposition tree of s_t is given by $\lceil \log_M (2t+2) \rceil$. Thus the size (number of threshold gates) of a recombiner computing $s_0, s_1, \ldots, s_R$ is given by

$$\sum_{t=0}^{R} \lceil (2t+1)/(M-1) \rceil, \tag{13}$$

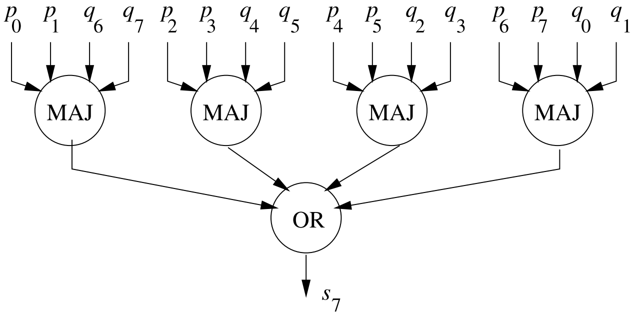
p_0 p_1 q_6 q_7 p_2 p_3 q_4 q_5 p_4 p_5 q_2 q_3 p_6 p_7 q_0 q_1
MAJ
MAJ
MAJ
MAJ
OR
s_7

Fig. 2. Decomposition of the majority function s_7 using majority and OR gates with fan-in bound of 4.

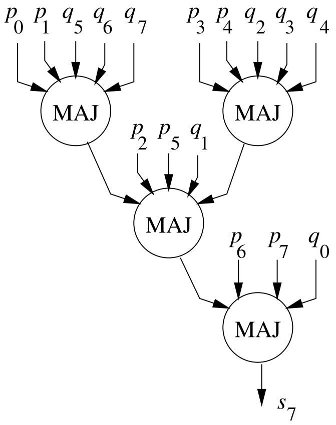

Fig. 3. Decomposition of the majority function s_7 using majority gates with fan-in bound of 5.

where M is the fan-in bound. The depth of the recombiner is governed by s_R, the most complex of its outputs and is given by $\lceil \log_M(2R+2) \rceil$.

Corollary 1 is also useful to implement the recombiner output s_t given by (10) when the fan-in bound M is odd. In this case, the sum over $t+1$ values of index i is partitioned into several sums, each over $\lceil M/2 \rceil$ consecutive i values. Each of these sub-expressions is evaluated independently by first factoring from it p_i with the largest index, something that is possible because of (9). After removing p_i, the remaining sub-expression is a majority function because of Corollary 1(b). This allows one to replace the complete sub-expression by a product of p_i with the output of a majority function. Thus the number of variables in the expression of s_t is greatly reduced, but the form of the expression and the relationships between the remaining variables are similar to the original expression of s_t. Thus this procedure can be applied recursively till a complete realization of s_t is achieved.

The procedure described above is best illustrated by an example. Consider the computation of s_7 using majority gates with a fan-in bound $M=5$. In this case,

$$\begin{aligned} s_7 &= (p_0q_7+p_1q_6+p_2q_5)+(p_3q_4+p_4q_3+p_5q_2)+p_6q_1+p_7q_0 \\ &= p_2(p_0q_7+p_1q_6+q_5)+p_5(p_3q_4+p_4q_3+q_2)+p_6q_1+p_7q_0 \\ &= p_2u_1+p_5u_0+p_6q_1+p_7q_0, \end{aligned} \tag{14}$$

where $u_1=p_0q_7+p_1q_6+q_5$ and $u_0=p_3q_4+p_4q_3+q_2$ are majority functions because of Corollary 1(b). Note that the factoring of p_2 from the first three product terms is based on expressing p_0 and p_1 as $p_0=p_0p_2$ and $p_1=p_1p_2$ using (9). Factoring of p_5 from the second sub-expression uses the same reasoning. It is easy to see that $q_1 \leqslant u_0 \leqslant u_1$. Thus the expression (14) has the same form as the original expression s_7 and the variables involved also follow similar conditions, namely, $p_2 \leqslant p_5 \leqslant p_6 \leqslant p_7$ and $q_0 \leqslant q_1 \leqslant u_0 \leqslant u_1$. Employing a similar procedure to implement (14), one gets

$$\begin{aligned} s_7 &= p_6(p_2u_1+p_5u_0+q_1)+p_7q_0 \\ &= p_6v_0+p_7q_0, \end{aligned} \tag{15}$$

where $v_0=p_2u_1+p_5u_0+q_1$ is a majority function (Corollary 1(b)). The expression (15) again has the same form as the original expression for s_t and the variables involved satisfy $p_2 \leqslant p_5$ and $q_0 \leqslant v_0$. Thus (15) is a majority function (Corollary 1(b)). This decomposition of s_7 is shown in Fig. 3.

The decomposition of s_t into threshold gates with an odd fan-in M is attractive because it uses only majority gates. However, these decompositions do not have the optimum depth $\lceil \log_M(2t+2) \rceil$ as in the case of an even M.

The discussion above is summarized as the following theorem.

Theorem 4 *(Recombiner output implementation). Each output s_t defined by (10) can be implemented using a multilevel network of generalized majority functions with any given bound on the fan-in.*

The total size of the network proposed in Fig. 1 is obtained by adding the number of gates in $\lceil n/M \rceil$ fragments to those in $\lceil n/M \rceil - 1$ recombiners. Each fragment uses $(R+1)$ threshold gates with a maximum fan-in of M. Thus level 0 of the decomposition (the fragments) has $\lceil n/M \rceil (R+1)$ threshold gates and a depth of 1.

The number of gates used in each recombiner is given in (13) and equals

$$\text{Recombiner size} = \sum_{t=0}^{R} \bigl\lceil (2t+1)/(M-1) \bigr\rceil. \tag{16}$$

We will estimate the complexity of a recombiner by computing (16) exactly in those cases when $(R+1)$ is a multiple of $(M-1)$ and (n/M) is a power of 2. Under this assumption, for the $(M/2)$ values of t satisfying $i(M-1) \leqslant t < i(M-1) + (M/2)$, the summation argument in (16) reduces to $2i+1$. Similarly for the $(M/2)-1$ values of t satisfying $i(M-1) + (M/2) \leqslant t < (i+1)(M-1)$, it reduces to $2i+2$. Therefore one can rewrite (16) as

$$\begin{aligned} \text{Reco. size} &= (M/2) \sum_{i=0}^{(R+1)/(M-1)-1} (2i+1) + \bigl((M/2)-1\bigr) \sum_{i=0}^{(R+1)/(M-1)-1} (2i+2) \\ &= (R+1)\bigl(R+(M/2)\bigr)/(M-1). \end{aligned} \tag{17}$$

Note that there are $(n/M)2^{-l}$ recombiners in level l, $1 \leqslant l \leqslant \log_2(n/M) - 1$. The recombiner in level $\log_2(n/M)$ computes only one output, s_R, and therefore has a size $\lceil (2R+1)/(M-1) \rceil = 2(R+1)/(M-1)$. Thus the size G of the network (i.e., the total number of threshold gates in the implementation) is obtained as:

$$\begin{aligned} G &= (n/M)(R+1) + \left[2(R+1) + \sum_{l=1}^{\log_2(n/M)-1} (R+1)\bigl(R+(M/2)\bigr)(n/M)2^{-l} \right] \Big/ (M-1) \\ &= (n/M)(R+1) + \left(\frac{R+1}{M-1}\right)\left(\frac{M}{2}+R\right)\left(\frac{n}{M}-2\right) + \frac{2(R+1)}{M-1}. \end{aligned} \tag{18}$$

Expression (18) shows that the size of the network is $O(nR^2/M^2)$.

The depth of a network is defined as the maximum length of the path from the root to a leaf. In the case of the network modeled by Fig. 1, this path passes through a fragment and several levels of recombiners. The maximum path through a fragment is of length 1 since it passes through only one gate (Theorem 1), while the maximum path through a fragment is the path from its inputs to its output s_R which is a function of $2(R+1)$ inputs. To obtain this output with gates with a fan-in bound M requires an M-ary tree of $\lceil \log_M(2R+2) \rceil$ (for even M) levels (Fig. 2). The depth D of the network can therefore be obtained by adding the depth of the fragment to the depth of $\log_2(n/M)$ recombiners to give

$$D = 1 + \log_2(n/M) \bigl\lceil \log_M(2R+2) \bigr\rceil. \tag{19}$$

Thus the network depth has order $O(\log(n/M) \log R / \log M)$.

If the function being decomposed is in $\widehat{LT}_1$, then R is $O(n^\delta)$ for some constant δ. Thus the decomposition size and depth will be $O(n^{2\delta+1}/M^2)$ and $O(\log^2 n / \log M)$ respectively. If the function being decomposed is instead in LT_d, then one can first convert it into $\widehat{LT}_{d+1}$ [6,21] with polynomial number of $\widehat{LT}_1$ functions. Each of these functions can then be decomposed using the strategy described here. The final decomposition will have a polynomial size and the same order of depth, $O(\log^2 n / \log M)$.

One should note that in many practical applications, the network size may be reduced considerably from the bounds presented here using the theorems given in Section 4.

4. Complexity reduction by redundancies in weight distribution

The decomposition presented in earlier sections partitions the inputs into sets, each applied to a different fragment. This section focuses on reducing the complexity of the network by making intelligent groupings of the inputs based on their weights. We show that the choice of the partition often affects the complexity of the fragments as well as that of the recombiners. In particular, if the *greatest common divisor* (gcd) of the input weights in a fragment is greater than 1, then some outputs of that fragment and its descendant recombiners may be redundant. Similarly, if the weights in a fragment are small compared with R, the fragment and its descendant recombiners may have an output redundancy.

Even though the weight constraints discussed in this section may appear artificial, they arise naturally in many real applications such as the ones discussed in Section 5.

We explore two kinds of redundancies in the fragment and the recombiner outputs. Let the output of a fragment or a recombiner be denoted by p_i, $0 \leqslant i \leqslant R$. We often refer to this entire sequence of outputs simply by p. When every k consecutive p_is are the same irrespective of the input as in (20) below,

$$p_{kt+a} = p_{kt}, \quad 0 \leqslant a < k, \quad \text{for all } t. \tag{20}$$

we say that p has a *block redundancy* of k.

The second kind of redundancy arises because of the relationship between p_is given in (8). This equation implies that when some $p_i = 1$, all subsequent p_is are 1 as well. When $p_i = 1$ for all $i \geqslant B$ irrespective of the input, we say that p *is bounded by* B.

Theorems in this section explore the conditions under which such redundancies occur.

Theorem 5 *(Fragment redundancy-I). If the weights in a fragment have the greatest common divisor of g, then its output p has a block redundancy of g.*

Proof. Let input x_i of fragment j have weight w_i. Then from the definition (7) of p_i, p_{gt+a}, $0 \leqslant a < g$, is equal to 1 if and only if

$$K_j - \sum_{i=1}^{N} x_i w_i \leqslant gt + a, \tag{21}$$

where K_j, being the sum of all the positive weights of the fragment, is a multiple of g. Thus, Eq. (21) can be written as

$$(K_j/g) - \sum_{i=1}^{N} x_i (w_i/g) \leqslant t + (a/g). \tag{22}$$

Since all the terms except the last in the inequality (22) are integers, (22) is equivalent to

$$\left(K_j - \sum_{i=1}^{N} x_i w_i\right) \Big/ g \leqslant t,$$

showing that p_{gt+a} is independent of a. □

Note that when the output p has a block redundancy of g, the only outputs one needs to compute are p_{gt}, $0 \leqslant t < (R+1)/g$. Each output p_{gt} is a threshold function with threshold $K_j - gt$, where K_j is the sum of all the positive weights of the fragment. Thus all the weights in the fragment as well as its threshold are multiples of g and consequently p_{gt} can be implemented as a threshold function with weights (w_i/g) and a threshold of $(K_j/g - t)$. Hence, the conditions of Theorem 5 not only imply fewer threshold functions, but also threshold functions with smaller weights. To illustrate this theorem, consider a fragment with inputs $x_1, \ldots, x_4$ with weights 2, -2, 4 and -6. Since the greatest common divisor of the weights is 2, the outputs of the fragment can be shown to be

$$p_{2t+1} = p_{2t} = \mathbf{TH}(x_1, x_2, x_3, x_4;\ 1, -1, 2, -3;\ 3 - t).$$

The second kind of redundancy shows up in the fragment output when the weights of inputs to a fragment are small relative to the critical error. In this case, the fragment contributes only small errors leading to the bounds on its output.

Theorem 6 *(Fragment redundancy-II). Let w_i, $i \in S_j$, denote the weights of inputs to the jth fragment. Then the output of the fragment is bounded by $\sum_{i \in S_j} |w_i|$.*

Proof. The error E_j of the jth fragment satisfies

$$E_j = \sum_{\substack{i \in S_j \\ w_i > 0}} w_i - \sum_{i \in S_j} x_i w_i \leqslant \sum_{i \in S_j} |w_i|.$$

The result then directly follows from the definition of p_t. □

Theorem 6 is important in reducing the complexity of a fragment that has small weights in relation to weights of the other fragments. To illustrate this theorem, once again consider the fragment with weights 2, -2, 4 and -6. The output of this fragment is bounded by 14, i.e., output $p_i = 1$, for all $i \geqslant 14$ irrespective of the input. Thus, no matter how large R is, one need not compute these p_is.

The next three theorems allow us to reduce recombiner complexity using input redundancies.

Theorem 7 *(Recombiner redundancy-I). If the input p of a recombiner has a block redundancy of g, then the recombiner outputs can be expressed as*

$$s_t = \sum_{i=0}^{\lfloor t/g \rfloor} p_{gi} q_{t-gi}, \tag{23}$$

where q is its other input.

Proof. The recombiner output is given by

$$s_t = \sum_{j=0}^{t} p_j q_{t-j}. \tag{24}$$

Let index $j = gi + i'$, $0 \leqslant i' < g$. Because p has a block redundancy of g, $p_j = p_{gi}$. Eq. (24) can then be rewritten as

$$s_t = \sum_{i=0}^{\lfloor t/g \rfloor} p_{gi} \left(\sum_{i'=0}^{g-1} q_{t-gi-i'} \right). \tag{25}$$

In (25) several q_is are being added (using Boolean OR). But since q_is satisfy (8), their sum equals the q_i amongst them with the largest index. In (25) this corresponds to the smallest i', i.e., q_{t-gi}. □

Note that s_t expressed as in (23) is still a majority logic function because of Corollary 1(a). However, as shown in Theorem 7, the block redundancy g of its input reduces it from a majority function of $2t + 2$ variables to a majority function of only $2\lfloor t/g \rfloor + 2$ variables. Clearly, if both inputs of a recombiner have block redundancies, one should use the larger of the two redundancies to reduce the recombiner complexity to the maximum extent possible. When the two block redundancies have a common factor, the recombiner complexity can be further reduced as stated in the following theorem.

Theorem 8 *(Recombiner redundancy-II). Let inputs p and q of a recombiner have block redundancies of g_1 and g_2 respectively. Then its output s has a block redundancy of $g = \gcd(g_1, g_2)$.*

Proof. We will show that the recombiner output s_{gt+a}, $0 \leqslant a < g$, is independent of a. Since, the block redundancy g_1 of p is a multiple of g, Theorem 7 gives

$$s_{gt+a} = \sum_{i=0}^{t} p_{gi} q_{g(t-i)+a}. \tag{26}$$

The block redundancy g_2 of q, also being a multiple of g gives

$$q_{g(t-i)+a} = q_{g(t-i)}, \quad 0 \leqslant a < g. \tag{27}$$

Combining (26) and (27) shows that s_{gt+a} is independent of a. □

Theorems 7 and 8 play an important role in minimizing the recombiner architecture when both its inputs have block redundancies. Theorem 8 shows that one only needs to compute every gth output of such a combiner where g is the gcd of the two block redundancies. Theorem 7 shows that the architecture for each of these outputs can be reduced by a factor equal to the larger of the two redundancies. Thus if the two inputs of a recombiner have block redundancies of 4 and 6, the recombiner size complexity can be reduced by approximately a factor of 12.

We next explore the bounds on the output of a recombiner.

Theorem 9 *(Recombiner redundancy-III). Let p and q denote the inputs of a recombiner and g, the block redundancy of its output. Also, let $p_{gi} = 1$ if $i \geqslant P$ and $q_{gi} = 1$ if $i \geqslant Q$. Then the output of the recombiner satisfies*

$$s_{gt} = 1, \quad \textit{if } t \geqslant P + Q. \tag{28}$$

Proof. From the definition of a recombiner output, one has

$$s_{gt} = \sum_{i=0}^{gt} p_i q_{gt-i}. \tag{29}$$

If $t \geqslant P + Q$, then there exists an index $i = gP$ in (29) such that $gt - i \geqslant gQ$. For this i, $p_i = 1$ and $q_{gt-i} = 1$ implying that at least one term in the Boolean summation (29) is 1. This reduces the entire sum to 1. □

Note that Theorems 6 and 9 together imply that a recombiner output is bounded by the sum of absolute values of the weights within all the fragments of whom it is a descendant.

Finally we present a theorem that exploits the input block redundancies together with their bounds.

Theorem 10 *(Recombiner redundancy-IV). Let inputs p and q of a recombiner have block redundancies g_1 and g_2 respectively with $g = \gcd(g_1, g_2)$. Further, let p and q be bounded such that $p_{ug} = 1$ for $u \geqslant P$ and $q_{vg} = 1$ for $v \geqslant Q$. Then, the output of the recombiner is given by*

$$s_{gt} = p_{(t-Q)g} + q_{\lfloor (t-P)g/g_2 \rfloor g_2} + \sum_{i=\lfloor (t-P)g/g_2 \rfloor + 1}^{Qg/g_2 - 1} p_{gt-ig_2} q_{ig_2}. \tag{30}$$

Proof. From Theorem 7 one gets

$$s_{gt} = \sum_{i=0}^{\lfloor gt/g_2 \rfloor} p_{gt-ig_2} q_{ig_2}. \tag{31}$$

We shall evaluate (31) by partitioning index i into three non-overlapping ranges.

If $i \geqslant Qg/g_2$, one has $q_{g_2 i} = 1$ from the bound on q. Thus, for this range of i values, the summation in (31) reduces to

$$\sum_{i=Qg/g_2}^{\lfloor gt/g_2 \rfloor} p_{gt-ig_2} = p_{(t-Q)g}. \tag{32}$$

Similarly, for $i \leqslant \lfloor (t-P)g/g_2 \rfloor$, one has $gt - ig_2 \geqslant Pg$ and therefore $p_{gt-ig_2} = 1$. For this range of i, the summation in (31) reduces to

$$\sum_{i=0}^{\lfloor (t-P)g/g_2 \rfloor} q_{ig_2} = q_{\lfloor (t-P)g/g_2 \rfloor g_2}. \tag{33}$$

Finally, for the remaining i values, the product $p_{gt-ig_2} q_{ig_2}$ will have to be summed as in (31). □

One should note that terms p_i and q_i in (30) are considered valid only if their index i is non-negative. Thus even though the theorem gives a general expression for s_{gt}, for specific values of t, some of the terms in the expression may be absent.

5. Examples

To illustrate the decomposition and complexity reduction methodology developed here, we now provide three examples, namely the decomposition of the majority function, the error tolerant pattern matching function and the comparison function. Let N denote the number of inputs. For the first of these examples, the critical error is about $N/2$, for the second, it is generally very small compared to N, and for the third, it increases exponentially with N.

5.1. Decomposition of a majority function

Majority and generalized majority functions occur in many applications. A monotonically increasing symmetric function is a generalized majority function. It is also known that any n variable symmetric function can be implemented in a two layered structure of at most $n+1$ generalized majority functions [1]. Further, any threshold logic function can be implemented in a three level network of generalized majority functions [12,20]. Finally, Theorem 2 in this paper shows that any arbitrary threshold function can be decomposed into a two level network with bounded fan-in threshold functions at the first level and a generalized majority function at the second level.

Recall that an n input majority function is a threshold function with unit weights of all the inputs and a threshold of exactly $\lfloor n/2 \rfloor + 1$. A generalized majority function of inputs $x_1, \ldots, x_n$ with a threshold $T > \lfloor n/2 \rfloor + 1$ can always be realized as a majority function with $2T - 2$ inputs $x_1, \ldots, x_n$, $0, \ldots, 0$ and a threshold T. Similarly, a generalized majority function with a threshold $T < \lfloor n/2 \rfloor + 1$ can be realized as a majority function with $2n - 2T + 1$ inputs $x_1, \ldots, x_n$, $1, \ldots, 1$ and a threshold $n - T + 1$. We therefore focus on the decomposition of a majority rather than a generalized majority function in this subsection.

Theorem 11 *(Decomposition of a majority function). A majority function with N inputs can be decomposed into a network of threshold functions with maximum fan-in of M such that the network size is $O(N^2/M)$ and the network depth is $O(\log^2 N/\log M)$.*

Proof. For convenience we will assume that $N = 2^n$ and $M = 2^m$ ($m < n$). Because the threshold in this application is $(N/2) + 1$, the critical error $R = (N/2) - 1$.

We follow the decomposition strategy described in Section 2. As in Theorem 6, the output of each fragment is bounded by M. Clearly, one needs to compute only M outputs from each of the (N/M) fragments. Thus, level 0 of the decomposition (comprising of fragments) has a size of N and a depth of 1.

The inputs p and q to recombiners in level 1 are the outputs of the fragments and are therefore bounded by M. Further, since all the weights in the majority function are 1, the greatest common divisor of any subset of these, $g = 1$. Theorem 9 then implies that the outputs of the recombiners in level 1 are bounded by $2M$. Since these outputs become inputs to the recombiners in level 2, Theorem 9 shows that the outputs of these recombiners are bounded by 2^2M. Proceeding in this manner, it is clear that the inputs of a recombiner in level l, $1 \leqslant l \leqslant n - m - 1$, are bounded by $P = 2^{l-1}M$ and its output is bounded by $2P$. Since $2P \leqslant R + 1$, one only needs to compute outputs s_0 through s_{2P-1} of this recombiner. Consider now the computation of a typical output s_t, $0 \leqslant t < 2P$ of a recombiner in level l. Because of the bounds on the inputs of the recombiner, the function s_t can be expressed from Theorem 10 as

$$s_t = p_{t-P} + q_{t-P} + \sum_{i=t-P+1}^{P-1} p_{t-i}q_i. \tag{34}$$

The p_is and q_is in (34) are valid only if their indices are non-negative. Thus the number of terms $p_{t-i}q_i$ in the summation in (34) equals

$$\min\{P - 1, t\} - \max\{t - P + 1, 0\} + 1.$$

Further, terms p_{t-P} and q_{t-P} contribute to s_t in (34) only if $t \geqslant P$. Thus,

$$\text{The total number of inputs of } s_t = \begin{cases} 2(t+1) & \text{if } 0 \leqslant t < P, \\ 2(2P - t) & \text{if } P \leqslant t < 2P. \end{cases} \tag{35}$$

Eq. (35) shows that the number of inputs to s_{2P-1-t} is the same as the number of inputs of s_t, $0 \leqslant t < P$. The complexity of s_{2P-1-t} and s_t being the same, the size (number of gates) G_l of a recombiner at level l, $1 \leqslant l \leqslant n - m - 1$, is obtained by summing the number of gates in each s_t as:

$$G_l = 2\sum_{t=0}^{P-1} \lceil (2t + 1)/(M - 1) \rceil, \tag{36}$$

where $P = 2^{l-1}M$.

Note that the argument of the summation in (36) evaluates to 1 for the first $M/2$ values of t, 2 for the next $M/2 - 1$, 3 for the next $M/2$, 4 for the following $M/2 - 1$, etc. Clearly, for $i(M - 1) \leqslant t < i(M - 1) + (M/2)$, the summation argument is $2i + 1$ and for $i(M - 1) + (M/2) \leqslant t < (i + 1)(M - 1)$, it is $2i + 2$. To take advantage of this, first define $K(l)$ as

$$\begin{aligned} K(l) &= \bigl(P - P \bmod (M - 1)\bigr)/(M - 1) \\ &= \bigl(2^{l-1}M - 2^{(l-1) \bmod m}\bigr)/(M - 1). \end{aligned} \tag{37}$$

The complexity of each recombiner G_l in level l, $1 \leqslant l \leqslant n - m - 1$, can now be rewritten as

$$\begin{aligned} G_l &= 2\left(2^{(l-1) \bmod m}\bigl(2K(l) + 1\bigr) + \sum_{i=1}^{2K(l)} (M/2)i - \sum_{i=1}^{K(l)} 2i\right) \\ &= (2M - 2)K^2(l) + (M - 2)K(l) + 2^{(l-1) \bmod m}\bigl(4K(l) + 2\bigr). \end{aligned} \tag{38}$$

Depth $d(l)$ of the recombiner in level l, $1 \leqslant l \leqslant n - m - 1$, is given by

$$d_l = \lceil \log_M 2^l M \rceil. \tag{39}$$

The recombiner in level $n - m$ has to compute only s_R. Since each parent of this recombiner provides exactly $2^{n-m} = (R + 1)$ outputs, s_R has $(2R + 2)$ inputs. Its size is therefore $\lceil (2R + 1)/(M - 1) \rceil = \lceil (N - 1)/(M - 1) \rceil$ threshold gates and its depth is $\log_M(2R + 2)$.

The total size G of the decomposition is therefore given by

$$G = N + \lceil (N - 1)/(M - 1) \rceil + \sum_{l=1}^{n-m-1} 2^{n-m-l} G_l. \tag{40}$$

By combining (37), (38) and (40) and simplifying, one gets

$$G = N + \left\lceil \frac{N - 1}{M - 1} \right\rceil + \frac{N}{2(M - 1)}\bigl(N + M(n - m) + \delta\bigr), \tag{41}$$

Table 1
Size of our decomposition of an N-input majority function into threshold functions with fan-in $\leqslant M$. Results of [29,30] are shown in parenthesis for comparison.

No. of inputs N	Fan-in M						
	2	4	8	16	32	64	128
4	7						
	(7)						
8	31	11					
	(31)	(13)					
16	127	49	19				
	(511)	(77)	(25)				
32	511	195	89	35			
	(16 383)	(2493)	(217)	(49)			
64	2047	753	353	169	67		
	(1.05E6)	(1.60E5)	(13 945)	(689)	(97)		
128	8191	2915	1347	673	329	131	
	(1.34E8)	(2.0E7)	(1.79E6)	(88 305)	(2401)	(193)	
256	32 767	11 377	5145	2561	1313	649	259
	(3.43E10)	(5.23E9)	(4.57E8)	(2.26E7)	(614 881)	(8897)	(385)

where δ gives the less significant terms as

$$\delta = M\left(\frac{m}{M-1} - 3\right) - 2(n-m) + c\left(\frac{(2-m)M^2-2}{M(M-1)} + (n-1) \bmod m\right) - \frac{Nc^2}{M^2}$$

and

$$c = 2^{-m\lfloor (n-m-1)/m \rfloor}.$$

Eq. (41) shows that this decomposition has a size $O(N^2/M)$.

The depth d of the decomposition can be obtained by finding the maximum length of a path from an input to the final output. This path goes through a fragment and $n-m$ recombiners. The depth d is therefore obtained as

$$d = 1 + \sum_{l=1}^{n-m} \lceil \log_M 2^l M \rceil = 1 + (n-m) + \sum_{l=1}^{n-m} \lceil \log_M 2^l \rceil. \tag{42}$$

Note that for all the ls satisfying $im < l \leqslant (i+1)m$, the argument of the summation in (41) becomes $(i+1)$. Thus the network depth may be simplified as

$$\begin{aligned} d &= 1 + (n-m) + \sum_{l=1}^{K} lm + (n-m-Km)(K+1) \\ &= 1 + (K+2)(n-m) - K(K+1)m/2, \end{aligned} \tag{43}$$

where $K = \lfloor (n-m)/m \rfloor$. Thus the decomposition depth is of order $O(n^2/m)$. □

Two values of M merit further discussion. When $M = 2$, the only threshold functions used in the decomposition are the 2-input Boolean AND and OR functions. Eqs. (41) and (43) show that in this case, the decomposition size is $N^2/2 - 1$ and the depth is $n(n+1)/2$. When $M = N/2$, the decomposition size is $N + 3$ gates and its depth is 3.

Previous work on symmetric functions has shown that the size of a symmetric function implemented by (2-input) AND and OR is $O(N^2)$ gates [28]. Note however that monotonically increasing symmetric functions are generalized majority functions and 2-input Boolean ANDs and ORs are threshold functions. Thus Theorem 11 can be considered as a generalization of [28] to multiple input threshold gates.

The results of this subsection can be directly compared with the earlier work on majority decomposition reported in [29,30]. The decompositions obtained in [29,30] have a size of $O(N^{\log N})$ and a depth of $O(\log^2(N/M))$ whereas our decompositions have a size $O(N^2/M)$ and a depth of $O(\log^2 N/\log M)$. The size and depth of our decomposition given by (41) and (42) are compared with those of [29,30] in Tables 1 and 2.

5.2. Threshold logic for error tolerant pattern matching

In many quality control and robotics applications, one has to compare a pattern captured by sensors with a stored template. In most of these applications the comparison needs to allow for a certain number of sensor errors. It is known that this problem of *error tolerant pattern matching* for binary patterns can be solved by a single threshold logic circuit [13, 31]. Let binary vectors $\mathbf{x}$ and $\mathbf{y}$ denote the input and the template respectively. The weight vector is created from $\mathbf{y}$ by

Table 2
Depth of our decomposition of an N-input majority function into threshold functions with fan-in $\leqslant M$. Results of [29,30] are shown in parenthesis for comparison.

No. of inputs N	Fan-in M						
	2	4	8	16	32	64	128
4	3						
	(3)						
8	6	3					
	(6)	(3)					
16	10	5	3				
	(10)	(6)	(3)				
32	15	8	5	3			
	(15)	(10)	(6)	(3)			
64	21	11	7	5	3		
	(21)	(15)	(10)	(6)	(3)		
128	28	15	10	7	5	3	
	(28)	(21)	(15)	(10)	(6)	(3)	
256	36	19	13	9	7	5	3
	(36)	(28)	(21)	(15)	(10)	(6)	(3)

replacing all the zeros in it by -1s. The threshold is chosen to be $wt(\mathbf{y}) - \epsilon$, where ϵ is the error tolerance and $wt(\mathbf{y})$ denotes the weight of the $\mathbf{y}$. For example, the threshold function

$$\mathbf{TH}(\mathbf{x};\ 1, 1, 1, -1, -1, 1, -1, 1;\ 2)$$

will output a 1 if the 8-bit input vector $\mathbf{x}$ matches with the pattern $\langle 1, 1, 1, 0, 0, 1, 0, 1\rangle$ with three or less errors.

In most applications, the number of inputs to this threshold function may get very large, rendering the threshold function impractical. In such cases, the methods of this paper can be used to decompose the function into smaller threshold functions as described by the following theorem.

Theorem 12 *(Decomposition of the error tolerant pattern matching function). The error tolerant pattern matching threshold function with N inputs and an error tolerance of ϵ can be decomposed into a network of threshold functions with maximum fan-in of M such that*

1. *the network size is $O(\epsilon N/M)$ and the network depth is $O(\log(N/M))$ for small ϵ satisfying $2(\epsilon + 1) \leqslant M$ and*
2. *the network size is $O(\epsilon^2 N/M^2)$ and the network depth is $O(\log(N/M)\log\epsilon/\log M)$ for larger ϵ.*

Proof. In the case of this threshold function, the sum of positive weights of inputs equals $wt(\mathbf{y})$ where $t(\mathbf{y})$ is the target pattern and the threshold equals $wt(\mathbf{y}) - \epsilon$. Thus the critical error $R = \epsilon$. As shown in Fig. 1, the decomposition uses (N/M) fragments and $(N/M) - 1$ recombiners.

Except for the last recombiner which has only one output, all the other recombiners as well as the fragments have $R + 1$ outputs, each of which is a function of at most $2(R + 1)$ inputs. Thus if $2(R + 1) \leqslant M$, then each of these outputs can be computed by a single threshold gate with fan-in M. Therefore the network has a size of $(2(N/M) - 2)(R + 1) + 1$ threshold gates and a depth of $1 + \log_2(N/M)$.

When $M < 2(R + 1)$, an output from a recombiner is not computable by a single threshold gate with a fan-in of M. One may then use the fact that the outputs of the fragments are bounded by M because of Theorem 6. Further, because of Theorem 9, the outputs of recombiners in level l are bounded by $2^l M$ as long as $2^l M \leqslant R + 1$. One can thus compute the complexity of the recombiners in these levels in a manner similar to the one used to determine the complexity of a majority gate in Section 5.1. However, this complexity is still bounded above by the complexity obtained without using Theorem 9. We therefore can use the general expressions in (18) and (19) to express this bound on the complexity. □

In Theorem 12 we have opted not to apply complexity reduction Theorem 9 when $M < 2(R + 1)$. This may be justified by noting that the number of levels to which this complexity reduction applies depends on the value of ϵ relative to M and could be a small number for realistic values of ϵ and M. For example, consider a 128-bit error tolerant pattern matching circuit with 5% error tolerance of 7 bits realized using threshold gates with a fan-in bound 4. One can see that in this case, the complexity reduction is not possible in any recombiner level.

Decomposition of an error tolerant pattern matching network is shown in Fig. 4. We assume a 32-bit input vector $\mathbf{x}$ is being matched with a 32-bit pattern with an error tolerance $\epsilon = 1$. The threshold functions A_i and B_i in Fig. 4 represent the outputs p_0 and p_1 from fragment i and are given by

$$\begin{aligned} &\mathrm{A}_i\colon\quad \mathbf{TH}(x_{4i+3}, \ldots, x_{4i};\ w_{4i+3}, \ldots, w_{4i};\ K_i), \\ &\mathrm{B}_i\colon\quad \mathbf{TH}(x_{4i+3}, \ldots, x_{4i};\ w_{4i+3}, \ldots, w_{4i};\ K_i - 1), \end{aligned} \tag{44}$$

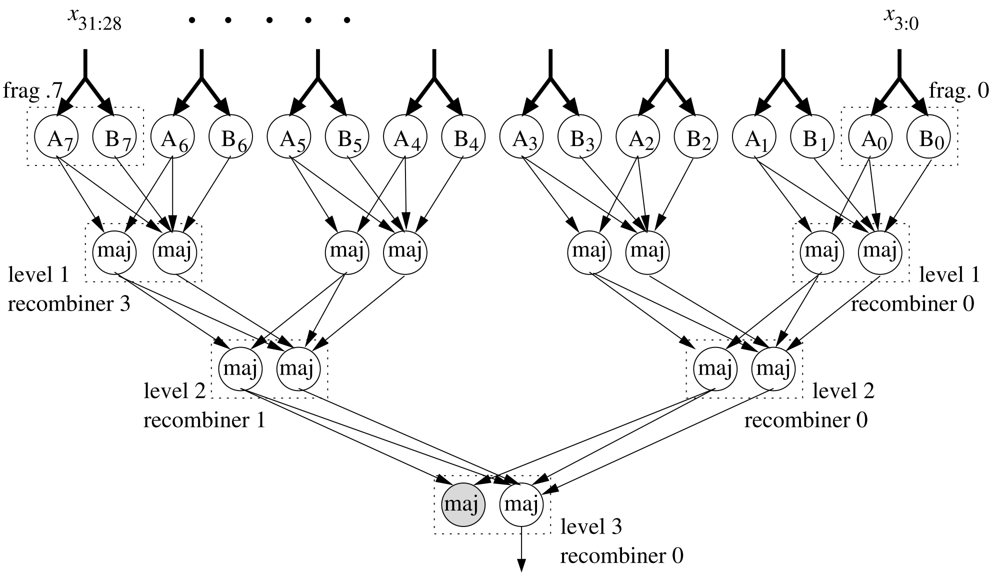

Fig. 4. The decomposition of a 32-bit pattern matching threshold function with error tolerance of 1 bit into a network of threshold functions with fan-in of 4 or less. A$_i$ and B$_i$ are threshold functions given in (44) and the rest are majority functions. Note that the shaded threshold function need not be implemented.

where $w_j = 1$ if the jth bit of the target pattern is 1, and -1 otherwise. K_i is the number of 1s in $\langle w_{4i+3}, \ldots, w_{4i}\rangle$. Threshold functions in each recombiner provide outputs s_0 and s_1 and are therefore majority functions as shown in Theorem 3.

5.3. Threshold logic for comparison

Two N-bit numbers $\mathbf{x} = \langle x_{N-1}, \ldots, x_2, x_1, x_0\rangle$ and $\mathbf{y} = \langle y_{N-1}, \ldots, y_2, y_1, y_0\rangle$ may be compared by the threshold function

$$\mathbf{TH}(\mathbf{x}, \mathbf{y};\ \mathbf{w}, -\mathbf{w};\ 0), \tag{45}$$

where vector $\mathbf{w} = \langle 2^{N-1}, \ldots, 2^2, 2, 1\rangle$. The output of this threshold function is 1 if $\mathbf{x} \geqslant \mathbf{y}$.

One can see that the comparison threshold function (45) is not a member of $\widehat{LT}_1$ since its weights increase exponentially with n. In fact, it is often used to show that $\widehat{LT}_1$ is a proper subset of LT_1. This function has attracted quite a bit of attention [12,19] because the number of inputs and the weights in this function get rather large with an increase in n. To compare two 32-bit numbers as in (45), one needs a threshold function with 64 inputs and weights as large as 2^{31}. In what follows, we show that the methods of this paper allow one to decompose (45) in a variety of ways.

Without loss of generality, assume that the number of bits, $N = 2^n$. We partition the inputs such that every $M = 2^m$ consecutive bits of $\mathbf{x}$ and $\mathbf{y}$ are applied to a fragment, i.e., the jth fragment, $0 \leqslant j < (N/M)$ has inputs x_{jM+i}, y_{jM+i}, $0 \leqslant i < M$. (Fan-in bound of each gate is assumed to be $2M$.) For the threshold function of (45), the critical error $R = 2^N - 1$. Thus a decomposition as in Fig. 1 should require 2^N outputs from each fragment and recombiner. However, using the results of Section 4 one can show that only two outputs from each fragment and recombiner are sufficient as shown in Fig. 5. Blocks A and B of fragment j in this figure are threshold functions with fan-in bound of 8 and are given by

$$\begin{aligned} &\text{A:}\quad \mathbf{TH}(x_{4j+3}, x_{4j+2}, x_{4j+1}, x_{4j}, y_{4j+3}, y_{4j+2}, y_{4j+1}, y_{4j};\ 8, 4, 2, 1, -8, -4, -2, -1;\ 0),\\ &\text{B:}\quad \mathbf{TH}(x_{4j+3}, x_{4j+2}, x_{4j+1}, x_{4j}, y_{4j+3}, y_{4j+2}, y_{4j+1}, y_{4j};\ 8, 4, 2, 1, -8, -4, -2, -1;\ 1). \end{aligned} \tag{46}$$

The rest of the functions in the decomposition are majority functions.

Alternately, one can decompose the same comparison function (45) into a network similar to that of Fig. 5, but with 16 fragments and 15 recombiners (arranged in 4 levels), using threshold gates with a fan-in bound of 4. Fragment j of this network will have two 4-input threshold functions defined as:

$$\begin{aligned} &\text{A:}\quad \mathbf{TH}(x_{2j+1}, x_{2j}, y_{2j+1}, y_{2j};\ 2, 1, -2, -1;\ 0),\\ &\text{B:}\quad \mathbf{TH}(x_{2j+1}, x_{2j}, y_{2j+1}, y_{2j};\ 2, 1, -2, -1;\ 1). \end{aligned}$$

The recombiners will have the same majority gate structure as in Fig. 5. This new design uses 57 gates and has a depth of 5 as against 26 gates and a depth of 4 of the design in Fig. 5. Thus, our decomposition strategy allows a trade-off between the fan-in bound, the hardware complexity (number of gates) and the time complexity (implementation depth).

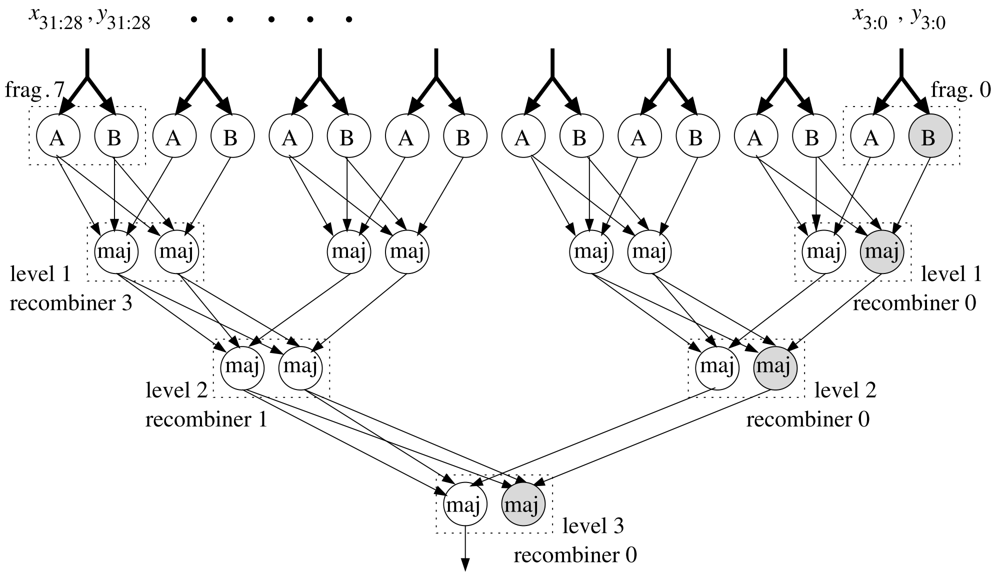

Fig. 5. The decomposition of a 32-bit comparison threshold function into a network of threshold functions with fan-in bound of 8. Functions A and B are given in (46) and the rest are majority functions. Note that the threshold functions along the right edge (shaded) need not be implemented.

The correctness of the decomposition shown in Fig. 5 follows from the following theorem.

Theorem 13 *(Decomposition of the comparison function). The $(N = 2^n)$-bit comparison threshold function can be decomposed into a network of threshold functions with fan-in bound of $2M$ for any $M = 2^m$, $1 < m \leqslant n$. This network has a size $4(N/M) - \log_2(N/M) - 3$ and a depth $d = \log_2(N/M) + 1$.*

Proof. Let the levels of the network be numbered from 0 to $n - m$, where level 0 corresponds to the fragments. Let the fragments and recombiners at any level be indexed in ascending order with those with index 0 corresponding to the lowest weight input bits. (See Fig. 5.)

The jth fragment has weights 2^{jM+i} and -2^{jM+i}, $0 \leqslant i < M$. Since these weights have a greatest common divisor of 2^{jM}, the output of this fragment has a block redundancy of 2^{jM} (Theorem 5). Similarly, by using induction (over the level l) one can show that the output of the jth recombiner in level l has a block redundancy of 2^{jM2^l} (Theorem 8). For mathematical convenience, we will henceforth denote the quantity $2^{M2^{l-1}}$ by G_l. G_l has the following property which would be used later.

$$G_l = (G_{l-1})^2. \tag{47}$$

In this new notation, the output of the jth recombiner in level l has a block redundancy of $(G_l)^{2j}$.

The outputs of some fragments and recombiners are also bounded. In particular, the output of the jth fragment is bounded as $p_{gt} = 1$ if $t \geqslant 2(2^M - 1)$, where $g = 2^{jM}$ represents the block redundancy of the fragment (Theorem 6). Similarly, by using induction over l, the output of the jth recombiner in the lth level can be shown to be bounded as $s_{gt} = 1$ if $t \geqslant 2((G_l)^2 - 1)$, where $g = (G_l)^{2j}$ is its block redundancy (Theorem 9).

We now use mathematical induction on level l to prove the following statement.

(S) The only outputs s_t required from the jth recombiner in the lth level are for $t = g((G_l)^2 - 1)$ for any j and $t = g((G_l)^2 - 2)$ for $j \neq 0$, where g is the block redundancy of the output of that recombiner.

At level $l = n - m$, there is only one recombiner with index $j = 0$. The only output required from this recombiner is s_R, where $R = 2^N - 1$. For this recombiner, $g = 1$ and $G_{n-m} = 2^{M2^{n-m-1}} = \sqrt{2^N}$. Thus statement **(S)** is true for $l = n - m$.

Now assume that **(S)** is true for the jth recombiner in level $l \geqslant 2$. We prove its validity for its parents in level $l - 1$ with indices $2j$ and $2j + 1$. Let p_i and q_i denote their outputs. Their block redundancies are $g_1 = (G_{l-1})^{4j} = (G_l)^{2j}$ and $g_2 = (G_{l-1})^{4j+2} = g_1 G_l$. Note that Theorem 8 can be used to relate the block redundancy, g, of the recombiner output to the block redundancies of its parents' outputs as $g = \gcd(g_1, g_2) = g_1$.

Because of the assumption, the only outputs required from the jth recombiner in level l are s_{gt_1}, where $t_1 = (G_l)^2 - 1$ and, if $j \neq 0$, also s_{gt_2}, where $t_2 = (G_l)^2 - 2$. We use Theorem 10 to obtain these values. The bounds on the outputs p_i and q_i of the parents are specified as $p_{g_1 t}, q_{g_2 t} = 1$ if $t \geqslant 2(G_{l-1}^2 - 1) = 2(G_l - 1)$. But the block redundancy, g, of the recombiner output is related to the block redundancies of the parents' outputs as $g_1 = g$ and $g_2 = gG_l$. Therefore the bounds on the two outputs may be expressed (in the language of Theorem 10) as $P = 2(G_l - 1)$ and $Q = 2G_l(G_l - 1)$.

To compute s_{gt_1}, note that $t_1 - Q < 0$ and $\lfloor (t_1 - P)g/g_2 \rfloor g_2 = (G_l - 2)g_2$. Therefore Theorem 10 gives:

$$\begin{aligned} s_{gt_1} = s_{g((G_l)^2-1)} &= q_{(G_l-2)g_2} + \sum_{i=G_l-1}^{2G_l-3} p_{((G_l)^2-1-iG_l)g_1} q_{ig_2} \\ &= q_{(G_l-2)g_2} + p_{(G_l-1)g_1} q_{(G_l-1)g_2}. \end{aligned} \tag{48}$$

Note that the summation in the above equation has only one valid term for $i = G_l - 1$ because for all higher values of i, the index of p becomes negative.

Similarly, one can show that if $j \neq 0$, then the required s_{gt_2} can be computed as

$$s_{gt_2} = q_{(G_l-2)g_2} + p_{(G_l-2)g_1} q_{(G_l-1)g_2}. \tag{49}$$

Eqs. (48) and (49) show that the parent with output p_i and index $2j$ needs to provide $p_{g_1 t}$ for $t = G_l - 1 = (G_{l-1})^2 - 1$, and also $t = G_l - 2 = (G_{l-1})^2 - 2$ if $j \neq 0$. The same equations also show that the parent with output q_i and index $2j+1$ needs to provide $q_{g_2 t}$ when $t = G_l - 1 = (G_{l-1})^2 - 1$ and $t = G_l - 2 = (G_{l-1})^2 - 2$. Thus statement **(S)** is true for recombiners in level $l - 1$ if it is true for recombiners in level l. From mathematical induction, the statement **(S)** is therefore true for all the recombiners in levels $1 \leqslant l \leqslant n - m$. Thus every recombiner except those with index 0 has only two outputs. Further, from Eqs. (48) and (49) and Theorem 3, these outputs are 3-input majority functions. Thus, each recombiner uses 2 gates if its index is nonzero and 1 otherwise. Depth of every recombiner is 1.

Since the recombiners in the 1st level need only two inputs from each of its parents, each fragment also needs to compute only two outputs. In particular, the jth fragment needs to compute outputs with indices $(2^M - 1)g$ and $(2^M - 2)g$ where g is its block redundancy. From Theorem 1, these outputs can be computed by threshold functions with thresholds $K_j - (2^M - 1)g$ and $K_j - (2^M - 2)g$, where K_j is the sum of all the positive weights within that fragment. Since $K_j = (2^M - 1)g$, the two threshold functions in the fragment have thresholds 0 and g. By scaling all the weights and threshold values by g, one can see that the two threshold functions in each fragment will have weights 2^i and -2^i, $0 \leqslant i < M$ and thresholds 0 and 1. Thus fragments with nonzero index have a size 2 and the one with index 0 has a size 1.

The network size as stated in the theorem can be obtained by adding sizes of (N/M) fragments and $(N/M) - 1$ recombiners. Similarly, the stated depth can be obtained by adding the depth of a fragment to the depth of one recombiner in each level. □

As shown in [19], decomposing the N-bit comparison threshold function to minimize the weights while keeping the depth small results in a constant depth 2 network, but the network size increases to $O(N^4 \log N)$. Further, some of the threshold functions in the network of [19] have a fan-in as large as $2N$. In contrast, the decomposition of the same comparison function given in Theorem 13 has a size $O(N/M)$ where M is some small number. About half of the threshold functions in this decomposition have a fan-in of $2M$ and maximum weight of 2^{M-1}; The rest are 3-input majority functions. The decomposition uses only three different threshold functions and therefore may be attractive for implementation. It should however be noted that this low fan-in, low weights and low size has been achieved at the cost of the network depth which has grown to $O(\log(N/M))$.

6. Conclusions

This paper has focused on decomposing any function in $\widehat{LT}_1$ into a polynomial size, $\log^2$ depth network of threshold functions with bounded fan-in. However, since $LT_d \subseteq \widehat{LT}_{d+1}$, the results here can also be used to decompose any function in LT_d for a constant d into threshold functions with bounded fan-in. This decomposition will also have polynomial size and $\log^2$ depth. Further, our explicit construction of the network allows one to trade-off the size and depth of the network with the fan-in bound.

Allowing the use of arbitrary threshold functions in the decomposition has helped us reduce the network size substantially as compared to the decompositions in NC^1. For example, the classical decomposition restricts the gates in all but the first level to AND and OR. As a result, this bounded fan-in decomposition of a function with n-inputs yields a network of size $O(2^n)$ and depth $O(n)$. This is strikingly different from our polynomial size and $O(\log^2 n)$ depth. Similarly, for the decomposition of a majority function into bounded fan-in gates, Refs. [29,30] use only ANDs in the middle levels and ORs in the top levels. As a result, their decomposition size is $O(n^{\log n})$; whereas allowing arbitrary threshold gates at all levels, we get a decomposition size $O(n^2)$.

The combinatorial relationships among the input weights of a threshold function may be exploited using the properties obtained here to further reduce our decomposition complexity. We demonstrate this by showing, for example, that an n-bit comparison function can be decomposed into bounded fan-in threshold gates using a network of size $O(n)$ and depth $O(\log n)$. This reduction in complexity is in spite of the fact that the comparison function is in LT_1 and not in $\widehat{LT}_1$.

Acknowledgments

The authors would like to thank the anonymous reviewers for helpful comments that have greatly improved the quality of this manuscript.

References

[1] S. Muroga, Threshold Logic and Its Applications, Wiley–Interscience, New York, 1971.

[2] V. Beiu, J.M. Quintana, M.J. Avedillo, VLSI implementations of threshold logic – a comprehensive survey, IEEE Trans. Neural Netw. 14 (5) (2003) 1217–1243.

[3] C. Pacha, K. Goser, Design of arithmetic circuits using resonant tunneling diodes and threshold logic, in: Proc. of the 2nd Workshop on Innovative Circuits and Systems for Nanoelectronics, Delft, NL, 1997, pp. 83–93.

[4] C. Lageweg, S. Cotofana, S. Vassiliadis, A linear threshold gate implementation in single electron technology, in: Proc. IEEE-CS Annual Workshop on VLSI, Orlando, FL, 2001, pp. 93–98.

[5] I. Amlani, A.O. Orlov, G. Toth, G.H. Bernstein, C.S. Lent, G.L. Snider, Digital logic gate using quantum-dot cellular automata, Science 284 (5412) (1999) 289–291.

[6] M. Goldmann, M. Karpinski, Simulating threshold circuits by majority circuits, SIAM J. Comput. 27 (1) (1998) 230–246.

[7] E. Allender, Circuit complexity before the dawn of the new millennium, in: Lecture Notes in Computer Science, vol. 1180, Springer-Verlag, 1996, pp. 1–18.

[8] A. Maciel, D. Thérien, Threshold circuits of small majority-depth, Inform. and Comput. 146 (1) (1998) 55–83.

[9] S. Cotofana, S. Vassiliadis, Signed digit addition and related operations with threshold logic, IEEE Trans. Comput. 49 (3) (2000) 193–207.

[10] W. Hesse, E. Allender, D. Barrington, Uniform constant-depth threshold circuits for division and iterated multiplication, J. Comput. System Sci. 65 (2002) 695–716.

[11] K.-Y. Siu, V. Rowchowdhury, On optimal depth threshold circuits for multiplication and related problems, SIAM J. Discrete Math. 7 (1994) 284–292.

[12] N. Alon, J. Bruck, Explicit construction of depth-2 majority circuits for comparison and addition, SIAM J. Discrete Math. 7 (1) (1994) 1–8.

[13] V. Annampedu, M.D. Wagh, Approximate pattern matching in nanotechnology, in: Proc. of Nanotech 2006, vol. 3, Boston, MA, 2006, pp. 316–319.

[14] Y. Leblebici, H. Özdemir, A. Kepkep, U. Çilingiroğlu, A compact high-speed $(31,5)$ parallel counter circuit based on capacitive threshold-logic gates, IEEE J. Solid-State Circuits 31 (8) (1996) 1177–1183.

[15] P. Mazumder, S. Kulkarni, M. Bhattacharya, J.P. Sun, G.I. Haddad, Digital circuit applications of resonant tunneling devices, Proc. IEEE 86 (4) (1998) 664–686.

[16] A. Hajnal, W. Maass, P. Pudlák, M. Szegedy, G. Turán, Threshold circuits of bounded depth, J. Comput. System Sci. 46 (1993) 129–154.

[17] J. Håstad, M. Goldmann, On the power of the small-depth threshold circuits, in: Computational Complexity 1, 1991, pp. 113–129.

[18] A.A. Razborov, On small depth threshold circuits, in: Scandinavian Workshop on Algorithm Theory, 1992, pp. 42–52.

[19] V. Bohossian, M. Riedel, J. Bruck, Trading weight size for circuit depth: An $\widehat{LT}_2$ circuit for comparison, Tech. Rep. Paradise, ETR028, California Institute of Technology, Nov. 1998.

[20] K.-Y. Siu, J. Bruck, On the power of threshold circuits with small weights, SIAM J. Discrete Math. 4 (3) (1991) 423–435.

[21] M. Goldmann, J. Håstad, A. Razborov, Majority gates vs. general weighed threshold gates, Comput. Complexity 2 (1992) 277–300.

[22] V. Beiu, H.E. Makaruk, Small fan-in is beautiful, in: Proc. of 1998 IEEE Int. Joint Conf. on Neural Networks, vol. 2, Anchorage, AK, 1998, pp. 1321–1326.

[23] P. Gupta, N.K. Jha, An algorithm for nano-pipelining of RTD-based circuits and architectures, IEEE Trans. Nanotechnol. 4 (2) (2005) 159–167.

[24] W. Prost, U. Auer, F.-J. Tegude, C. Pacha, K.F. Goser, G. Janssen, T. van der Roer, Manufacturability and robust design of nanoelectronic logic circuits based on resonant tunnelling diodes, Int. J. Circuit Theory Appl. 28 (2000) 537–552.

[25] G.S. Glinski, C.K. Yue, Decomposition of n-variable threshold function into p-variable threshold functions, where $p < n$, Tech. Rep. 63-10, Dept. of EE, Univ. of Ottawa, Canada, June 1963.

[26] V. Annampedu, M.D. Wagh, Building multi-input RTD circuits under reliability constraints, in: Proc. of the 2nd IEEE Int. Workshop on Defect and Fault Tolerant Nanoscale Architectures, Boston, MA, 2006, pp. 45–52.

[27] A. Yao, On ACC and threshold circuits, in: IEEE Symposium on Foundations of Computer Science (FOCS), 1990, pp. 619–627.

[28] O.N. Muzychenko, Uniform and regular structures for realization of symmetric functions of the algebra of logic, Autom. Remote Control 59 (4) (1998) 581–592.

[29] V. Beiu, J. Peperstraete, R. Lauwereins, Enhanced threshold gate fan-in reduction algorithms, in: ICYCS'93: Proceedings of the Third International Conference on Young Computer Scientists, Tsinghua University Press, Beijing, China, 1993, pp. 339–342.

[30] V. Beiu, J. Peperstraete, J. Vandewalle, R. Lauwereins, Overview of some efficient threshold gate decomposition algorithms, in: Proc. of 9th Intl. Conf. Control Systems and Comp. Sci. CSCS'93, Bucharest, Romania, 1993, pp. 458–469.

[31] V. Annampedu, M.D. Wagh, Reconfigurable approximate pattern matching architectures for nanotechnology, Microelectronics 38 (2007) 430–438.