Summary

This paper focuses on fanout-free networks of multivalued 2-input 1-output gates. A synthesis technique is demonstrated which is similar to the partition matrix approach used for binary networks. A special case of the fanout-free network, the cascade, is also considered. A recurrence relation for the number of cascade realizable functions is derived. It is shown that the addition of only one rail in a multi-rail binary cascade substantially increases the number of realized functions.

I. Introduction

Fanout-free networks of binary gates have received considerable attention recently (e.g., Hayes1, Maroukas and Honda3, Butler and Breeding3, Chakrabarti and Kolp7, and Kodandapant and Seth5). In such networks each gate output and each net input are applied only one gate input. Thus, the structure is a tree whose root node is a gate supplying the (single) network output. Since there is only one path from each input to the output, fault tests are easily implemented. Furthermore, fanout-free networks require fewer gates than non-fanout-free nets with the same number of inputs.

Networks of multivalued gates share these advantages, and it is the aim of this paper to present design techniques for such networks. The gates used have two inputs and one output. It will be assumed that all 2-variable m-valued functions are available, although, as it is shown later, not all gate types are necessary for the realization of every fanout-free function.

Also considered in this paper is the cascade (Maitra6, Yoeli8), a fanout-free network which consists of a single string of interconnected gates. The synthesis techniques are a special case of the techniques described for general fanout-free nets. Also, a recursion relation derived shows that an extremely large number of functions is realized by a cascade of only moderate size. A number of researchers (Short9, Yoeli10, and Sung11) have investigated multirail binary cascades, and it is well known that the theory of single-rail multivalued cascades is closely related to the theory of multirail binary cascades. This analogy is used to derive a recursion relation for the number of functions realized by multirail binary cascades.

II. Synthesis Technique for Multivalued
Fanout-Free Networks

The approach used here is an extension of the partition matrix technique applied by Ashenhurst12.* Supported in part by NSF Grant MCS-7600326.

In particular, let \( Z = \{z_1, z_2, \ldots, z_m\} \) be a set of \( m \)-valued variables and let \( f(Z) \) be an \( m \)-valued function on \( Z \). The values of \( z_i \) and \( f \) will be denoted \( 0, 1, \ldots, m-1 \). If \( f(Z) \) has a simple disjunctive decomposition (SDD) if and only if

\[
f(Z) = F(g(X), Y)
\]

where \( X \cup Y = Z \) and \( X \cap Y = \emptyset \). The SDD is nontrivial if \( |X| > 1 \) and \( |Y| > 1 \). \( f(Z) \) can be realized by two interconnected networks realizing \( g(X) \) and \( F(g, Y) \).

An important tool for the identification of SDD's is the partition matrix. Specifically, the \( Y \mid X \) partition matrix has \( m |X| \) columns and \( m |Y| \) rows labeled by all possible assignments of values to \( X \) and \( Y \), respectively. Each square contains the value of \( f \) for the assignment of values to \( X \) corresponding to the column labeling and for the assignment of values to \( Y \) corresponding to the row labeling. Fig. 1 shows the partition matrix of a 3-valued 4-variable function. The column (row) multiplicity \( \nu (\nu) \) is the number of distinct columns (rows) in the matrix. For this example, \( \nu = \nu = 3 \).

![Figure 1](image)

Figure 1. The \( z_3z_4|z_1z_2 \) Partition Matrix of \( f_1(z_1z_2, z_3z_4) \).
Theorem 1: \( f(Z) \), an \( m \)-valued function, has the SDD

\[
f(Z) = F(g(X), Y)
\]

if and only if its \( Y|X \) partition matrix has column multiplicity \( v \leq m \).

The 3-valued function shown in Fig. 1 satisfies Theorem 1 and thus has the SDD

\[
f_1(x_1, x_2, x_3, x_4) = F_1(g_1(x_1, x_2), x_3, x_4).
\]

(2)

The transpose matrix also satisfies the condition of Theorem 1 and therefore \( f_1 \) can be expressed as

\[
f_1(x_1, x_2, x_3, x_4) = F_2(g_2(x_3, x_4), x_1, x_2).
\]

(3)

However, the existence of two SDD's of the form (2) and (3) implies the existence of the complex disjunctive decomposition (CDD).

\[
f_1(x_1, x_2, x_3, x_4) = F_2(g_2(x_3, x_4), x_1, x_2)
\]

(4)

Eq. (4) can also be obtained from the following extension of Theorem 4.2 of Curtis-5.

Theorem 2: \( f(Z) \), an \( m \)-valued function, has the CDD,

\[
f(Z) = F(g_1(X), g_2(Y))
\]

(5)

if and only if its \( Y|X \) partition matrix has column multiplicity \( v \times m \) and row multiplicity \( u \leq m \).

Functions \( F \), \( g_1 \), and \( g_2 \) can be obtained directly from the \( Y|X \) partition matrix. Arbitrarily, assign to each of the \( u \) distinct rows a value 0, 1, ..., \( u-1 \), such that two different rows have different values. The columns are labeled in a similar manner. In the example of Fig. 1, choices for columns and row assignments are shown along the top and left side, respectively.

\( g_1 \), \( g_2 \), and \( F \) are then defined by these entries. For example, since \( z_1z_2 = 02 \) corresponds to \( 2 \), \( g_1(0,2) = 2 \). Since \( z_2z_4 = 21 \) corresponds to 0, \( g_2(2,1) = 0 \) and, since \( f_1(0,2,2,1) = 1 \), \( F_3(2,0) = 1 \). Fig. 2 shows the complete functions.

\[
\begin{array}{cccc|c|c|c}
  & 1 & 0 & 2 & 2 & 2 & 2 \\
 0 & 1 & 1 & 0 & 0 & 2 & 1 \\
 1 & 1 & 0 & 2 & 1 & 2 & 0 \\
 2 & 2 & 1 & 2 & 2 & 0 & 0 \\
\end{array}
\]

Figure 2. Partition Matrices of \( g_1, g_2, \) and \( F_3 \).

Since \( g_1g_2 \) and \( F_3 \) are realized by 2-input 1-output 3-valued gates, a fanout-free realization of \( f_1(z_1, z_2, z_3, z_4) \) has the form shown in Fig. 3.

Note that if \( g_1 \) and \( g_2 \) of (5) have missing logic levels, the value of \( F \) can be chosen arbitrarily for those levels (don't cares).

In the manner of the previous example, a synthesis algorithm for a fanout-free function \( f \) proceeds as follows:

1. Find all CDD's of \( F \), generating for each two subfunctions \( g_1(X) \) and \( g_2(Y) \). If \( |X| = 1 \) or 2, \( g_1(X) \) is fanout-free and no further test of \( g_1 \) is required. Similarly, if \( |Y| = 1 \) or 2, \( g_2(Y) \) is fanout-free and no further test is required.

2. Test all subfunctions generated from Step 1 for CDD's. Continue in this way until no CDD's are found (algorithm halts unsuccessfully) or all subfunctions depend on one or two variables (algorithm halts successfully).

As in the example, there will be a number of choices for the subfunctions (e.g., if \( v = 3 \), 2 or 1, there are \( 3!/2! \) 2 or 3 ways, respectively, to label columns). However, these choices will not affect fanout-free realizability, since a unary operation only is involved.

The synthesis algorithm is also a realizability test, since the algorithm halts unsuccessfully when no appropriate CDD's are found. In effect, the algorithm proceeds from the inputs to the output. An alternative algorithm proceeds from the inputs to the output. In particular, if a function \( f(Z) \) is fanout-free, it has at least one decomposition of the form

\[
f(Z) = H(g_1(z_1, z_2), 2 - [z_1, z_2])
\]

(6)

where \( H \) is fanout-free. Theorem 1 is again applicable and it follows that the \( 2 - [z_1, z_2] \) partition matrix has \( v \leq m \). It is then necessary to test \( H \) for decompositions of the form (6). In terms of the number of tests required, the latter algorithm has the advantage. That is, there are \( \binom{n}{2} \) different partition matrices to test initially where, for the first algorithm, there are \( 2^n(n-1) \) partition matrices.

However, as discussed later, there may be a significant reduction in both cases.

1There are a total of \( 2^n \) partition matrices, of which \( 2n + 2 \) are trivial (one or no variables in a row or column). Thus, there are \( 2^n-2n-2 \) nontrivial matrices. However, it is necessary to test only one half of these, because once a matrix is tested, there is no need to consider its transpose.
IV. Types of Gates Necessary in a Fanout-Free Realization

In these synthesis algorithms, it is assumed that all 2-input 1-output gates are available. However, specific functions can be eliminated without reducing the set of realizable fanout-free functions. In particular, let \( U = \{ u(z) \} \) be the set of all unary functions, and let \( R \) be a relation between functions \( f_1 \) and \( f_2 \) which depend on exactly 2 variables, such that \( f_1 \) is \( f_2 \) if and only if \( \sigma_1(x_1, x_2) = u_1(f_2(x_1, x_2), u_2(x_2)) \) or \( \sigma_1(x_1, x_2) = u_1(f_2(u_2(x_2), u_1(x_1))) \) for \( u_1, u_2, u_3 \in U \). \( R \) is an equivalence relation and, as such, induces a partition on the set of 2-variable functions. Partitions of this nature for binary circuits were studied by Slepian\(^6\) in his classic paper and more recently by Allen\(^7\) for multivalued circuits. If all unary operations are available, then only one representative of each equivalence class is required to realize all functions dependent on exactly two variables. An approximate upper bound on the number of representatives can be found as follows.

There are \( m^2 \), \( m^2 \), and \( m^2 \) functions on 2 or fewer variables, 1 or 0 variables, and 0 variables, respectively. Thus there are \( 2^{m^2} - 2(m^2-m) \) functions on exactly two variables. An upper bound on the number of representatives \( N(2,m) \) of 2-variable \( m \)-valued functions can be found by dividing \( a(m) \) by the number of elements in the smallest equivalence class. As far as is known this has not been calculated. Hence, we propose the following:

Conjecture 1: The smallest equivalence class in the partition induced by \( R \) on the set of two variable functions contains \( f(z_1, z_2) \) where

- \( f(i, i) = 1 \)
- \( f(i, j) = 0 \) for \( i \neq j \)

for \( 0 \leq i, j \leq m-1 \).

The \( z_1 \) \( z_2 \) partition matrix of \( f \) contains a diagonal of 1's; all other entries are 0. It is believed that \( f \) is in the smallest equivalence class because of the large number of permutations on values of \( z_1 \) and \( z_2 \) which result in the same function. For example, the function obtained from \( f \) by an interchange of values of \( z_1 \) can also be obtained by an interchange of values of \( z_2 \). For the case where \( m = 2 \) the smallest equivalence class contains two members, the exclusive OR and equivalence function. When \( m = 2 \) \( f(z_1, z_2) \) is the equivalence function.

A lower bound on the number of members of the class containing \( f \) is \( \binom{m}{2} 2^m \) for \( m > 2 \), \( \binom{m}{2} \) is the number of ways to make two choices from the values available. Except for \( m = 2 \), there will be more occurrences of one value than the other, and thus there are two ways to choose the more abundant one. Any permutation of values of \( z_1 \) or \( z_2 \) will produce only \( m \) different patterns. \( \binom{m}{2} 2^m \) is a lower bound since this count includes only unary functions \( u_1(z) \) which are permutations of values of \( z \). In the unary function \( u(z) = 0 \), for example, is not a permutation; it maps all values of \( z \) to 0.

Thus, an upper bound on the number of representatives of 2-variable \( m \)-valued functions \( N(2,m) \) is

\[
\binom{m^2}{2} 2^m, \quad m > 2
\]

(7)

Although this is a very loose upper bound, it does show that a significant reduction in the number of 2-input gates needed can be achieved.

V. Cascades of Multivalued Gates

A cascade is a fanout-free network which consists of a single chain of gates. Figure 4 shows a cascade of \( n \) 2-input 1-output gates. The synthesis algorithms for this case are an adaptation of those in the previous section. For example, in the first algorithm

\[
\begin{align*}
&\sigma_1(z_1, z_2) = f(1, 1) = 1, & i = j,
&\sigma_1(z_1, z_2) = f(1, 2) = 0, & i \neq j
\end{align*}
\]

Figure 4. A Cascade of \( n \) Multivalued Gates

The number of different arrangements obtained by all possible permutations of values of \( z_1 \) and \( z_2 \) is just the number of ways to place \( m \) non-taking rooks on an \( m \times m \) chessboard. This is \( m! \) (Liu18).

functions \( u_1(z) \) which are permutations of values of \( z \) (the unary function \( u(z) = 0 \), for example, is not a permutation; it maps all values of \( z \) to 0).

A special case of the multivalued cascade exists when \( m \), the number of logic levels, is an integral power of 2. If \( m = 2^t \), the cascade is equivalent to a cascade of binary gates with 1 outputs and 2 inputs. A number of other structures have considered cascades in which the number of binary lines between cells is different than the number of lines associated with the \( z \) inputs. This will be the assumption applied here. In particular, assume the set of inputs in Fig. 4 labeled \( z \) have \( r = 2^t \) values while the \( y \) inputs have \( n = 2^t \) inputs. Such a network is called a \( u \)-rail cascade. Fig. 5 shows this network. It will be convenient later to assume that the number of inputs applied to the leftmost gate from the left is \( t \) (the same number each gate receives from above). For the present, however, it is assumed that \( a = u \).

Figure 5. A \( u \)-rail Cascade.
VI. The Number of Functions Realized by Cascades

One measure of a particular network structure is the number of switching functions realized. Such measures indicate tradeoffs which can be made between the range of functions realizable and the complexity of the circuits. For example, intuition indicates that if the number of lines between cells is reduced, the set of functions realized at the output is also reduced. From an information theoretic view, fewer lines between gates means that less information about remote inputs can be passed to the cascade output. To put this in a more precise context, consider the following.

A cascade of \( n \) 2-input 1-output gates can be decomposed into a single gate driven by a cascade of \( n - 1 \) gates as shown in Fig. 6. The set of functions realized at \( y_n \) can be divided into \( s \) subsets, where subset \( S^s_1(s,r) \) consists of all functions which produce exactly \( i \) of \( s \) logic values, for \( 1 \leq i \leq s \). Subset \( S^s_1(s,r) \), for example, consists of functions whose output is one value (regardless of the values at the inputs). Let \( N_i(n,s,r) \) be the number of functions realized by a cascade of \( n \) gates, where \( y_0, y_1, \ldots, y_n \) take on \( s \) values and \( z_1, z_2, \ldots, z_n \) take on \( r \) values. Then,

\[
N_i(n,s,r) = \sum_{i=1}^{s} S^s_1(s,r)
\]

(8)

Since all 2-input gates are available one choice for \( G_n \) is a trivial one, producing one constant output. Thus, the cascade produces all trivial functions and

\[
|S^s_1(s,r)| = s
\]

The case where \( i > 1 \) can be treated by counting the number of ways to form the \( z_n | y_0z_1z_2 \cdots z_{n-1} \) partition matrix, \( M \), of a cascade realizable function. As is shown in Figure 7, this matrix has \( r \) rows and \( s^{n-1} \) columns.

\[
\text{function realized by } N \text{ (} s^{n-1} \text{ entries)}
\]

\[
\begin{array}{cccc}
  y_0 & z_1 & \cdots & z_{n-1} \\
  0 & 1 & \cdots & r-1 \\
\end{array}
\]

(9)

Let \( C_i(j,s,r) \) be the number of functions realized by a cascade of \( n \) gates which produces exactly \( j \) levels in which \( M \) has \( j \) columns. The summation ranges only to \( s \) because the number of columns

\[
|S^s_1(s,r)| = \sum_{j=1}^{s} C_i(j,s,r)
\]

(10)

\[1 \text{ in } M \text{ cannot exceed the number of logic levels appearing at } y_{n-1}.\]

If \( N \) realizes a \( j \) level function, \( f(y_0, z_1, z_2, \ldots, z_{n-1}) \), then it also realizes all functions obtained from it by an interchange of logic levels. This follows from the fact that \( G_{n-1} \) can be replaced by a gate realizing the same function as \( G_{n-1} \) except for an interchange of logic values. There are a total of \( \binom{s}{j} \) such functions including \( f \) itself. However, the set of the functions produced at \( y_n \) is the same for any of the \( \binom{s}{j} \) choices. Thus,

\[
C_i(j,s,r) = \frac{|S^s_1|}{\binom{s}{j}} \cdot D(1,j),
\]

(10)

where \( D(1,j) \) is the number of ways to assign \( i \) logic levels to partition matrix \( M \) such that all \( j \) columns are distinct. There are \( \binom{R}{j} \) ways to choose \( j \) logic levels. Given this choice there are \( \binom{R}{j} \) ways to form \( j \) distinct columns of \( r \) entries with \( i \) or fewer values. There are \( \binom{R-1}{j} \) ways to form \( j \) distinct columns with \( i \) or fewer values, etc. Applying the principle of inclusion/exclusion (pp. 95-106 Liu\textsuperscript{18}) yields

\[
\text{yield}
\]

42
\[
D(i,j) = \binom{s}{j} \sum_{k=0}^{i-1} (-1)^k \binom{i}{k} \binom{i-k}{j}\]

or
\[
D(i,j) = \binom{s}{j} \binom{i-1}{j-1} \binom{s}{j} \sum_{k=0}^{i-1} (-1)^k \binom{i}{k} \binom{i-k}{j}\]

from (9), (10), and (11) we obtain,
\[
|S_i^n(s,r)| = \sum_{j=1}^{s} \frac{|S_{i-1}^{n-1}(s,r)|}{\binom{s}{j}} \sum_{k=0}^{i-1} (-1)^k \binom{i}{k} \binom{i-k}{j}\]

(12) is a recurrence relation in which values of \( |S_i^n(s,r)| \) are expressed as a function of lower order values. The initial conditions are determined by the number of functions realized by the leftmost cell of the cascade. These are
\[
|S_1^n(s,r)| = \binom{s}{1} \sum_{k=0}^{r} (-1)^k \binom{r}{k} (1-k)^s r\]

(13)

|\(n\)| | \(N(n,s=2,r=2)\) | \(N(n,s=3,r=2)\) | \(N(n,s=4,r=2)\) |
|---|---|---|---|
|1 | 72 | 729 | 65,536 |
|2 | 88 | 47,601 | 77,575,936 |
|3 | 520 | 3,450,897 | 103,901,883,136 |
|4 | 3,112 | 252,034,065 | 1.39823 x 10^{14} |
|5 | 18,664 | 18,416,334,609 | 1.88194 x 10^{17} |
|6 | 111,976 | 1.34574 x 10^{12} | 2.53302 x 10^{20} |
|7 | 671,848 | 9.83380 x 10^{13} | 3.40934 x 10^{23} |
|8 | 4,031,080 | 7.18590 x 10^{15} | 4.58884 x 10^{26} |

|\(n\)| | \(N(n,s=2,r=3)\) | \(N(n,s=3,r=3)\) | \(N(n,s=4,r=3)\) |
|---|---|---|---|
|1 | 64 | 19,683 | 16,777,216 |
|2 | 1,744 | 53,267,787 | 9.34643 x 10^{12} |
|3 | 48,784 | 147,125,769,363 | 5.29473 x 10^{18} |
|4 | 1,365,904 | 4.06430 x 10^{14} | 2.99961 x 10^{24} |
|5 | 38,245,264 | 1.12275 x 10^{18} | 1.69936 x 10^{30} |
|6 | 1,070,867,344 | 3.10156 x 10^{21} | 9.62737 x 10^{35} |
|7 | 29,984,285,584 | 8.56795 x 10^{24} | 5.45418 x 10^{41} |
|8 | 839,559,996,304 | 2.36687 x 10^{28} | 3.08994 x 10^{47} |

|\(n\)| | \(N(n,s=2,r=4)\) | \(N(n,s=3,r=4)\) | \(N(n,s=4,r=4)\) |
|---|---|---|---|
|1 | 256 | 531,441 | 4,294,967,296 |
|2 | 30,496 | 44,307,654,561 | 7.20819 x 10^{17} |
|3 | 3,659,296 | 3.70573 x 10^{14} | 1.21222 x 10^{26} |
|4 | 439,115,296 | 3.10103 x 10^{10} | 2.03862 x 10^{44} |
|5 | 52,693,835,296 | 2.59430 x 10^{45} | 3.42860 x 10^{52} |
|6 | 6.32326 x 10^{12} | 2.17037 x 10^{30} | 5.76562 x 10^{50} |
|7 | 7.58791 x 10^{14} | 1.81572 x 10^{35} | 9.69619 x 10^{58} |
|8 | 9.10549 x 10^{16} | 1.51902 x 10^{40} | 1.63063 x 10^{67} |

Table I. The Number of Functions \(N(n,s,r)\) Realized by a Cascade of \(n\) Gates Where \(y_0,y_1,\ldots,y_n\) Take on \(s\) Values and \(x_1,x_2,\ldots,x_n\) Take on \(r\) Values.
The equations derived can be modified easily to yield the number of functions \( N_2(n, 2^u, 2^t) \) realized by each output of a u-rail cascade. Let \( f_1, f_2, \ldots, f_u \) denote these functions. Because of symmetry, the range of functions at each output is the same as any other. Furthermore, the range of functions at any output is as large as the range obtained from any switching function on \( f_1, f_2, \ldots, f_u \). That is, a function obtained by forming \( q = F(f_1, f_2, \ldots, f_u) \) is also realized by any function \( f_i \) since gate \( G_N \) can realize any function on its input variables. The number of functions at any output is then just the number of functions at \( y_n \) which produce two distinct values (say 0 and 1). This is

\[
N_2(n, 2^u, 2^t) = \frac{|S_2(n, t)|}{\binom{2^t}{2}} \big|_{t=2^u}.
\]

To determine how the number of binary functions realized by a u-rail cascade varies with \( s \), it will be assumed that for the initial gate, \( G_1 \), of Fig. 5, \( a = r \). Thus, changing the number of rails will not change the number of network inputs. In this way, we have a basis of comparison for determining how the number of rails affects the number of functions realized.

To account for the change, a new initial condition must be calculated. In particular, the number of multivalued functions realized by gate \( G_1 \) is now

\[
|S_1^1(r, s)| = \left(\frac{s}{t}\right) \sum_{k=0}^{t} (-1)^k \binom{t}{k} (1-k)^2
\]

Table II shows \( N_2(n, 2^u, 2^t) \) for various values of \( s \) and \( t \). Note that \( N_2(n, 2^u, 2^t) \) does not include the two trivial functions \( q = 0 \) and \( q = 1 \). It can be seen that the addition of one rail can substantially in-

<table>
<thead>
<tr>
<th>( n )</th>
<th>( N_2(n, s=2, t=1) )</th>
<th>( N_2(n, s=2, t=2) )</th>
<th>( N_2(n, s=2, t=3) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>14</td>
<td>65,534</td>
<td>1.84667 x 10^{19}</td>
</tr>
<tr>
<td>2</td>
<td>86</td>
<td>7,864,094</td>
<td>6.02102 x 10^{23}</td>
</tr>
<tr>
<td>3</td>
<td>518</td>
<td>943,691,296</td>
<td>1.96526 x 10^{28}</td>
</tr>
<tr>
<td>4</td>
<td>3,110</td>
<td>113,242,955,296</td>
<td>6.41461 x 10^{32}</td>
</tr>
<tr>
<td>5</td>
<td>18,662</td>
<td>1.35892 x 10^{13}</td>
<td>2.09373 x 10^{37}</td>
</tr>
<tr>
<td>6</td>
<td>111,974</td>
<td>1.63070 x 10^{13}</td>
<td>6.83393 x 10^{41}</td>
</tr>
<tr>
<td>7</td>
<td>671,846</td>
<td>1.95684 x 10^{13}</td>
<td>2.23059 x 10^{46}</td>
</tr>
<tr>
<td>8</td>
<td>4,031,078</td>
<td>2.34821 x 10^{13}</td>
<td>7.28067 x 10^{50}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( n )</th>
<th>( N_2(n, s=2, t=2) )</th>
<th>( N_2(n, s=2, t=3) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>14</td>
<td>65,534</td>
</tr>
<tr>
<td>2</td>
<td>254</td>
<td>7,52818 x 10^{12}</td>
</tr>
<tr>
<td>3</td>
<td>65,534</td>
<td>1.26584 x 10^{21}</td>
</tr>
<tr>
<td>4</td>
<td>77,575,934</td>
<td>2.12879 x 10^{29}</td>
</tr>
<tr>
<td>5</td>
<td>103,901,883,134</td>
<td>3.58004 x 10^{37}</td>
</tr>
<tr>
<td>6</td>
<td>1.39823 x 10^{14}</td>
<td>6.02065 x 10^{45}</td>
</tr>
<tr>
<td>7</td>
<td>1.88194 x 10^{17}</td>
<td>1.01251 x 10^{54}</td>
</tr>
<tr>
<td>8</td>
<td>2.53502 x 10^{20}</td>
<td>1.70276 x 10^{63}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( n )</th>
<th>( N_2(n, s=2, t=3) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>254</td>
</tr>
<tr>
<td>3</td>
<td>65,534</td>
</tr>
<tr>
<td>4</td>
<td>4,294,967,296</td>
</tr>
<tr>
<td>5</td>
<td>1.31700 x 10^{18}</td>
</tr>
<tr>
<td>6</td>
<td>2.09836 x 10^{27}</td>
</tr>
<tr>
<td>7</td>
<td>3.41827 x 10^{36}</td>
</tr>
<tr>
<td>8</td>
<td>5.56954 x 10^{45}</td>
</tr>
</tbody>
</table>

Tabel II. The Number of Nontrivial Functions \( N_2(n, s=2^u, t=2^t) \) Realized at a Single Output of a u-Rail Cascade Where \( n \) is the Number of Gates, \( u \) is the Number of Inputs From the Left, and \( t \) is the Number of Inputs From Above.
increase the number of functions realized. For example, consider a network of 4 gates where each gate receives one network input except the leftmost gate which receives two inputs. This network has a total of 5 inputs. For 1, 2 and 3 rails the cascade realizes 3,110, 77,575,934 and 4,294,967,294 nontrivial functions, respectively. Thus, the addition of just one rail substantially increases the number of functions which can be realized.

VII. Concluding Remarks

Because a search is required in the synthesis algorithms described, computational effort can be substantial for even a moderate number of inputs. It is reasonable to ask whether a reduction effort may be so restricted to a proper subset of equivalence class representatives. Hayes showed that in the binary case, if exclusive OR's are not used, an easily applied test for fanout-free realizations is possible. This may be true of multivalued gates, as well.

A reduction in computational effort appears to be possible even when no restriction is placed on the gate set. In particular,

Conjecture 2: If \( f(Z) \) has two CDD's

\[
f(Z) = H_1(g_1(X), g_2(Y))
\]

(16)

\[
f(Z) = H_2(g_3(W), g_4(Z))
\]

(17)

then \( g_1 \) and \( g_2 \) are fanout-free if and only if \( g_3 \) and \( g_4 \) are fanout-free.

If the conjecture is true, then there is no need to develop all CDD's of \( f(Z) \); the first one found is sufficient. A first step in the proof might be to show that (16) and (17) implies \( f(Z) \) also has the decomposition

\[
f(Z) = G(h_1(X \cap W), h_2(X \cap Z), h_3(Y \cap W), h_4(Y \cap Z)),
\]

a likely possibility in view of a closely related result for binary functions (Theorem 4.4, Curtis\(^{11}\)).

With respect to the second synthesis algorithm, a reduction in computational effort is possible if the following is true:

Conjecture 3: If \( f(Z) \) has the decomposition

\[
f(Z) = H_1(g_1(x_1, x_2), Z - \{x_1, x_2\})
\]

(18)

\[
f(Z) = H_2(g_2(x_k, x_t), Z - \{x_k, x_t\})
\]

(19)

then \( H_1 \) is fanout-free if and only if \( H_2 \) is fanout-free.

If true, then the search for partitions of \( f(Z) \) of the form (18) and (19) ends with the discovery of the first one.

With respect to the multi-rail cascade, Table II shows an interesting result. The data for \( N_s(n, s^2, r^2) \) indicate that each output of the three cascades in Figure 8 realize all nontrivial functions on the input variables. For \( n = 1 \), this is not surprising since \( G_1 \) can be any gate. For \( n = 2 \), since \( G_2 \) can produce at both outputs all functions on its input variables, and since the two outputs of \( G_1 \) can produce \( y_0 \) and \( z_1 \) at its output, all functions on \( y_0, z_1 \), and \( z_2 \) can appear at the output of \( G_2 \). On the other hand, for the three gate case and analogous situation does not exist, yet \( G_3 \) produces all functions! This is because all functions have the Shannon decomposition

\[
q_4 = f_3(y_0, z_1, z_2) + f_1(y_0, z_1, z_2)
\]

(20)

Since all pairs of functions on \( y_0, z_1 \), and \( z_2 \) can be realized at the output of \( G_3 \) if \( f_0 \) and \( f_1 \) of (20) can be specified as any desired function and thus \( q_4 \) can be any function on \( y_0, z_1, z_2 \). However, not all pairs of functions can be produced at the outputs of \( G_3 \). Thus, the inputs to gate \( G_0 \) if it exists, are not sufficient to cause the four gate network to produce all functions on its inputs. A similar situation exists for other values of \( u \) and \( t \), for example, for \( u = 3 \) and \( t = 1 \).

References


