A Design Method for Single-Rail LUT Cascades

Tsutomu Sasao,
Department of Computer Science,
Meiji University, Kawasaki, Japan

Abstract— This paper presents a method to realize logic functions by single-rail LUT cascades. Main results include: 1) Any $2m$-variable function can be realized by a single-rail cascade with $(m+1)$-LUTs. The number of LUTs is at most $2^{m+1}-1$. There exists a $2m+1$ variable function that cannot be realized by the single-rail cascade with $(m+1)$-LUTs. 2) When a $2m$-variable function has a functional decomposition $f(X_1, X_2)$, where X_1 and X_2 have m variables, and the column multiplicity of the decomposition is μ, the number of LUTs can be reduced to $2\mu-1$. 3) Any n-variable function can be realized by a single-rail cascade with seven $(n-1)$-LUTs. 4) Ad-hoc methods to realize a n-variable function using two or three $(n-1)$-LUTs.

I. Introduction

Two of the most crucial problems in modern LSIs are their long design time and short life cycles. A solution to these problems may be reconfigurable architecture. Reconfigurable architecture reduces the hardware development time drastically, since single LSI can be used for various applications.

Reconfigurable devices include random access memories (RAMs) and programmable logic arrays (PLAs). They are easy to design. However, when the number of input variables n is large, the necessary hardware becomes too large. Thus, field programmable logic arrays (FPGAs) are often used. FPGAs implement random logic networks of lookup tables (LUTs), and they require layout and routing in addition to logic design. Since the area for programming and interconnections are much larger than the logic area, FPGAs require large chip area. This paper considers the realization of logic functions by LUT cascades. Cascade is one of the simplest structures and easy to layout. Since the interconnection is limited to only the adjacent cells, area and delay are minimum.

Cascade realizations of logic functions were considered in 1960's [5]. Excellent survey papers can be found in [6].

The cascades can be classified as 1) **Single rail** and **multi-rail**, and 2) **Irredundant** and **redundant**[1]. Fig. 1.1 is a single-rail cascade with irredundant inputs; Fig. 1.2 is a multi-rail cascade with irredundant inputs; Fig. 1.3 is a single-rail cascade with redundant inputs; and Fig. 1.4 is a multi-rail cascade with redundant inputs.

In a single-rail cascade, the number of wires between adjacent cells is one, while in a multi-rail cascade, the number of wires is more than one [10]. In an irredundant cascade, each variable is connected to only one input terminal of the cascade, while in a redundant cascade, some variables are connected to two or more input terminals of the cascade. Irredundant cascades can represent only a fraction of functions. However, two-rail redundant cascades of three-input cells can represent any functions . A cell is also called a lookup table (LUT).

Efficient design methods for multi-rail cascades are available [12]. Both combinational and sequential implementations of multi-rail cascades exist. As for combinational realizations,

[1]This terminology was used in [14]. In this case, *redundant inputs* cannot be removed. So, **cascade with repeated input variables** may be more desirable terminology.

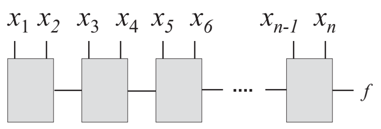

Fig. 1.1. Single-rail cascade with irredundant inputs.

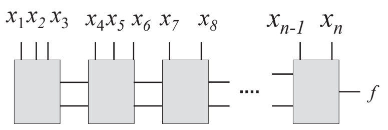

Fig. 1.2. Multi-rail cascade with irredundant inputs.

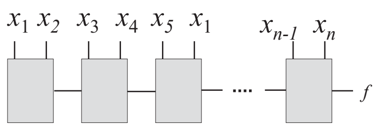

Fig. 1.3. Single-Rail cascade with redundant inputs.

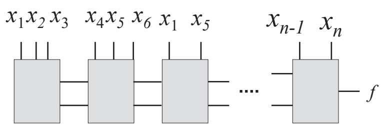

Fig. 1.4. Multi-Rail cascade with redundant inputs.

LSI chips for multi-rail cascades have been developed [8]. As for sequential realizations, both FPGA and emulator on a PC [7], were developed.

On the other hand, single-rail cascades were not available until recently. However, recently AMD developed FPGAs that can implement single-rail cascades [16]. They have special interconnections called **cascade multiplexer paths** among LUTs in a *slice* [16] of an FPGA, to minimize the interconnection delay. AMD Versal CLBs [16] have cascade multiplexer paths, which are useful for single-rail cascade. The delay times for an LUT and that for the routing between LUTs within the same CLB are in the range of 10 to 100 ps. On the other hand, the delay time for general routing highly depends on the length of the routing. For very short routing, it is around 300 ps, while for a long routing, it is around 3 to 7 ns. So, when the given function is realized by a single-rail cascade using at most eight LUTs, the delay time for the function can be estimated accurately.

When a function has an iterative disjoint decomposition, we can easily find an irredundant cascade using conventional decomposition algorithms [1, 2]. However, not all functions have such decompositions. As for redundant single-rail cascades, no efficient method to realize given functions were available to the authors' knowledge[2]. In this paper, we show a method to realize redundant single-rail cascades.

II. Systematic Method for Single-Rail Cascade

From here, we use the terminology in [9]. An LUT that realizes any n-input function is denoted by n-LUT.

Theorem 2.1 *Any function with $n = 2m$ variables can be realized by a single-rail cascade with $(m+1)$-LUTs. The number of LUTs is at most $2^{m+1} - 1$.*

(Proof) Expand the function $f(X_1, X_2)$ into

$$f(X_1, X_2) = \bigoplus_{i=0}^{2^m-1} X_1^{\vec{a}_i} f(\vec{a}_i, X_2),$$

where $X_1 = (x_1, x_2, \dots, x_m)$ and $X_2 = (x_{m+1}, \dots, x_n)$. This is obtained by applying Shannon expansions m times. $X_1^{\vec{a}_i} = 1$ if and only if $X_1 = \vec{a}_i$, and $\vec{a}_0 = (0, 0, \cdots, 0, 0)$, $\vec{a}_1 = (0, 0, \cdots, 0, 1)$, $\vec{a}_2 = (0, 0, \cdots, 1, 0)$, and $\vec{a}_3 = (0, 0, \cdots, 1, 1)$.

Consider the circuit shown in Fig. 2.1 ($m = 2$). It satisfies the relations:

$$\begin{aligned}
g_0 &= h_0(X_2), \\
g_1 &= g_0 \cdot \overline{p_1(X_1)} \oplus h_1(X_2), \\
g_2 &= g_1 \cdot \overline{p_2(X_1)} \oplus h_2(X_2), \\
\cdots &= \cdots \\
g_{2^m-1} &= g_{2^m-2} \cdot \overline{p_{2^m-1}(X_1)} \oplus h_{2^m-1}(X_2),
\end{aligned}$$

where $p_i(X_1) = 1$ iff $X_1 = \vec{a}_i$.

[2]An exception is [15], which is quite hard to read.

The circuit in Fig. 2.1 realizes $f(X_1, X_2)$ when

$$\begin{aligned}
h_{2^m-1}(X_2) &= f(\vec{a}_{2^m-1}, X_2), \\
h_{2^m-2}(X_2) &= f(\vec{a}_{2^m-2}, X_2) \oplus f(\vec{a}_{2^m-1}, X_2), \\
\cdots &= \cdots \\
h_1(X_2) &= f(\vec{a}_1, X_2) \oplus f(\vec{a}_2, X_2), \\
h_0(X_2) &= f(\vec{a}_0, X_2) \oplus f(\vec{a}_1, X_2).
\end{aligned}$$

It is clear that $h_i(X_2)$ for $i = 0, 1, \dots, 2^m - 1$ are unique. $\square$

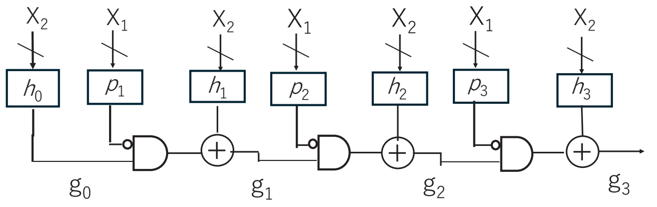

Fig. 2.1. Cascade realization for a function with $n = 2m = 4$ variables.

Example 2.1 Consider a function $f(X)$ with $n = 2m = 4$. In this case, partition X into $X_1 = (x_1, x_2)$ and $X_2 = (x_3, x_4)$. The function can be represented as

$$\begin{aligned}
f(X_1, X_2) &= \bar{x}_1\bar{x}_2 f_{00}(X_2) \oplus \bar{x}_1 x_2 f_{01}(X_2) \oplus \\
&\quad x_1\bar{x}_2 f_{10}(X_2) \oplus x_1 x_2 f_{11}(X_2).
\end{aligned}$$

Consider the cascade shown in Fig. 2.1.
In this case,

$$\begin{aligned}
p_0(X_1) &= \bar{x}_1\bar{x}_2, & p_1(X_1) &= \bar{x}_1 x_2, \\
p_2(X_1) &= x_1\bar{x}_2, & p_3(X_1) &= x_1 x_2.
\end{aligned}$$

- When $X_1 = (1, 1)$, the output of p_3 is 1. Thus, the right-most AND gate produces 0, and the output g_3 is

$$h_3(X_2) = f_{11}(X_2).$$

- When $X_1 = (1, 0)$, the output of p_2 is 1. Thus, the middle AND gate produces 0, and the output g_3 is

$$h_2(X_2) \oplus h_3(X_2) = f_{10}(X_2).$$

- When $X_1 = (0, 1)$, the output of p_1 is 1. Thus, the left-most AND gate produces 0, and the output g_3 is

$$h_1(X_2) \oplus h_2(X_2) \oplus h_3(X_2) = f_{01}(X_2).$$

- When $X_1 = (0, 0)$, all the outputs of p_i are 0. Thus, the signal propagates from the left to right, and the output g_3 is

$$h_0(X_2) \oplus h_1(X_2) \oplus h_2(X_2) \oplus h_3(X_2) = f_{00}(X_2).$$

From these equations, we have:

$$\begin{aligned} h_3(X_2) &= f_{11}(X_2), \\ h_2(X_2) &= f_{11}(X_2) \oplus f_{10}(X_2), \\ h_1(X_2) &= f_{10}(X_2) \oplus f_{01}(X_2), \\ h_0(X_2) &= f_{01}(X_2) \oplus f_{00}(X_2). \end{aligned}$$

■

Corollary 2.1 *Any function with $n = 2m + 1$ variables can be realized by a single-rail cascade with $(m + 2)$-LUTs. The number of LUTs is at most $2^{m+1} - 1$.*

(Proof) In the proof of Theorem 2.1, let $X_1 = (x_1, x_2, \ldots, x_m)$ and $X_2 = (x_{m+1}, \ldots, x_n)$, where X_2 has $m + 1$ variables. In this case, LUTs for h_i has $(m + 2)$ inputs. The number of LUTs is 2^{m+1}. □

The circuit derived from the proof of Theorem 2.1 require many LUTs. When the given function has a functional decomposition, we can realize a cascade with fewer LUTs.

Theorem 2.2 *Any function with $n = 2m$ variables can be realized by a single-rail cascade with $(m + 1)$-LUTs. The number of LUTs is at most $2\mu - 1$, where μ is the column multiplicity of the decomposition chart for $f(X_1, X_2)$, where $|X_1| = |X_2| = m$.*

(Proof) Suppose that the function $f(X_1, X_2)$ can be represented as

$$f(X_1, X_2) = \sum_{i=0}^{\mu-1}\!\!\oplus\, p_i(X_1) q_i(X_2),$$

where $p_i(X_1)$ shows the columns that produce the column function $q_i(X_2)$ $(i = 0, 1, 2, \cdots, \mu - 1)$ as shown in Fig. 2.2.

$$\begin{array}{c|c|c|c|c|} & \multicolumn{4}{c}{X_1} \\ & p_0 & p_1 & \cdots & p_{\mu-1} \\ \hline X_2 & q_0 & q_1 & \cdots & q_{\mu-1} \\ \hline \end{array}$$

Fig. 2.2. Decomposition chart of a logic function with the column multiplicity μ.

Consider the circuit shown in Fig. 2.3 ($\mu = 3$). It satisfies the relations:

$$\begin{aligned} g_0 &= h_0(X_2), \\ g_1 &= g_0 \cdot \overline{p_1(X_1)} \oplus h_1(X_2), \\ g_2 &= g_1 \cdot \overline{p_2(X_1)} \oplus h_2(X_2), \\ \cdots &= \cdots \\ g_{\mu-1} &= g_{\mu-2} \cdot \overline{p_{\mu-1}(X_1)} \oplus h_{\mu-1}(X_2). \end{aligned}$$

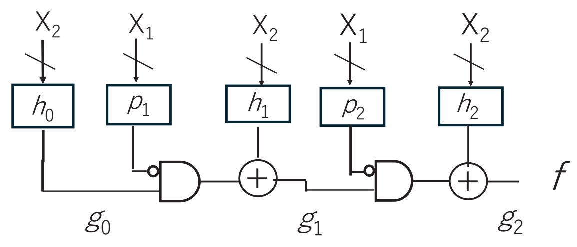

Fig. 2.3. Cascade realization for a function with column multiplicity $\mu = 3$.

This circuit (Fig. 2.3) realizes $f(X_1, X_2)$ when

$$\begin{aligned} h_{\mu-1}(X_2) &= f(|X_1 \in P_{\mu-1}), \\ h_{\mu-2}(X_2) &= f(|X_1 \in P_{\mu-2}) \oplus f(|X_1 \in P_{\mu-1}), \\ \cdots &= \cdots \\ h_1(X_2) &= f(|X_1 \in P_1) \oplus f(|X_1 \in P_2), \\ h_0(X_2) &= f(|X_1 \in P_0) \oplus f(|X_1 \in P_1), \end{aligned}$$

where P_i denotes the set of vectors that produces the column function $q_i(X_2)$ $(i = 0, 1, 2, \ldots, \mu - 1)$. □

Example 2.2 Consider the decomposition chart for the function $f(X_1, X_2)$ shown in Fig. 2.4. Note that the column multiplicity μ is three. Thus, the function $f(X_1, X_2)$ can be represented as

			X_1			
			0	0	1	1	x_1
			0	1	0	1	x_2
	0	0	1	0	0	0	
X_2	0	1	0	1	1	0	
	1	0	0	0	0	1	
	1	1	0	0	0	0	
	x_3	x_4	q_0	q_1		q_2	

Fig. 2.4. Decomposition chart of a logic function.

$$p_0(X_1)q_0(X_2) \oplus p_1(X_1)q_1(X_2) \oplus p_2(X_1)q_2(X_2),$$

where

$$\begin{aligned} p_0(X_1) &= \bar{x}_1\bar{x}_2, & q_0(X_2) &= \bar{x}_3\bar{x}_4, \\ p_1(X_1) &= x_1 \oplus x_2, & q_1(X_2) &= \bar{x}_3 x_4, \\ p_2(X_1) &= x_1 x_2, & q_2(X_2) &= x_3\bar{x}_4. \end{aligned}$$

Consider the circuit in Fig. 2.3. It satisfies the following relations:

$$\begin{aligned} g_0 &= h_0(X_2), \\ g_1 &= g_0 \cdot \overline{p_1(X_1)} \oplus h_1(X_2), \\ g_2 &= g_1 \cdot \overline{p_2(X_1)} \oplus h_2(X_2). \end{aligned}$$

From these, we have

$$\begin{aligned} h_2(X_2) &= q_2(X_2) = x_3\bar{x}_4, \\ h_1(X_2) &= q_1(X_2) \oplus q_2(X_2) = \bar{x}_3x_4 \oplus x_3\bar{x}_4, \\ h_0(X_2) &= q_0(X_2) \oplus q_1(X_2) = \bar{x}_3\bar{x}_4 \oplus \bar{x}_3x_4. \end{aligned}$$

We can confirm that the circuit in Fig. 2.3 realizes the function $f(X_1, X_2)$.

- When $X_2 = (1, 1)$, the output of p_2 is 1, so the right AND gate produces 0. So, the output g_2 is
$$h_2(X_2) = q_2(X_2) = x_3\bar{x}_4.$$
- When $X_2 = (1, 0)$ or $X_2 = (0, 1)$, the output of p_1 is 1, so the left AND gate produces 0. So, the output g_2 is
$$h_1(X_2) \oplus h_2(X_2) = q_1(X_2) = \bar{x}_3x_4,$$
- When $X_2 = (0, 0)$, all the outputs of p_i are 0. Thus, the signal pass through the cascade from the left to right, and the output g_2 is
$$h_0(X_2) \oplus h_1(X_2) \oplus h_2(X_2) = q_0(X_2) = \bar{x}_3\bar{x}_4.$$

■

Theorem 2.3 *There exists an $(n = 2m+1)$-variable function that cannot be realized by a single-rail cascade with $(m+1)$-LUTs using the method shown in Fig. 2.1.*

(Proof) An example for $m = 2$ is the majority function of five variables. It cannot be realized by a single-rail cascade with 3-LUTs. It can be verified by **lutexact**, a SAT based synthesis program, in *ABC* [4]. However, such method is only useful for the functions with a small number of inputs [13]. For larger n, we can prove as follows: Consider the function of $2m + 1$ variables that can be written as

$$f(X) = \bigoplus_{i=0} x_{i_1}x_{i_2}\cdots x_{i_{m+1}},$$

the EXOR sum of all the products with degree $m + 1$. Note that the circuit in Fig. 2.1 cannot produce the sum of all the products with degree $(m + 1)$. □

III. Design for Symmetric Functions

Theorem 3.1 *Any symmetric function with $2m$ variables can be represented by a single-rail cascade with $2m + 1$ LUTs of $(m + 1)$ inputs.*

(Proof) Assume that X is partitioned into $X_1 = (x_1, x_2, \ldots, x_m)$ and $X_2 = (x_{m+1}, \ldots, x_{2m})$. The column multiplicity of the decomposition chart of the symmetric function $f(X_1, X_2)$ is at most $\mu = m + 1$. Thus, from Theorem 2.2, the necessary number of LUTs to realize the function is at most $2m + 1$. □

Example 3.1 Consider the symmetric function of 6-variables:

$$f(X) = 1 \Leftrightarrow \sum_{i=1}^{6} x_i \geq 4.$$

In this case, since $m = 3$, the function can be realized with seven 4-LUTs. When the input variables are partitioned into $X_1 = (x_1, x_2, x_3)$ and $X_2 = (x_4, x_5, x_6)$. We have the reduced decomposition chart shown Fig. 3.1, where the labels show the numbers of 1's in X_1 and X_2. Thus, the function can be realized by the circuit shown in Fig. 2.1. Note that the column for q_0 shows the constant 0 function. So, if q_0 and q_3 are exchanged, the rightmost LUT can be removed. Thus, the number of necessary LUTs can be reduced to 6. By the way, the **lutexact** produced a solution with four 4-LUTs, which is exact minimum. ■

			X_1		
		0	1	2	3
	0	0	0	0	0
X_2	1	0	0	0	1
	2	0	0	1	1
	3	0	1	1	1
		q_0	q_1	q_2	q_3

Fig. 3.1. Reduced decomposition chart for a 6-variable symmetric function.

Theorem 3.2 *Any symmetric function of 8 variables can be realized by a single-rail cascade with seven 6-LUTs.*

(Proof) Partition the inputs X into $X_1 = (x_1, x_2, x_3)$ and $X_2 = (x_4, x_5, x_6, x_7, x_8)$. Since the column multiplicity is at most $\mu = 4$, the number of LUTs is $2\mu - 1 = 7$. Also, the number of inputs to an LUT is at most 6.

By the way, the *lutexact* command in *abc* produced solutions with at most four 4-LUTs. □

IV. Ad-hoc Methods to Derive Single-Rail Cascades

The design method in the previous sections is systematic. That is, it always derives a circuit if a solution exists. Unfortunately, it often requires more LUTs than necessary. In this part, we show **ad-hod methods**, which can quickly derive a solution with fewer LUTs.

Theorem 4.1 *Any n-variable function can be realized by a single-rail cascade with seven $(n - 1)$-LUTs, when $n \geq 5$.*

(Proof) Partition the inputs X into $X_1 = (x_1, x_2)$ and $X_2 = (x_3, x_4, \ldots, x_n)$. Since the column multiplicity is at most $\mu = 4$, the number of LUTs is $2\mu - 1 = 7$. □

Note that Theorem 4.1 cannot be used recursively.

In the following theorems, we can find more efficient circuits in special cases. Unfortunately, it is not systematic, so we may miss a solution, even it exists.

Theorem 4.2 *Let f be a n-variable function. Suppose that f is written as*

$$f(x_1, X_2) = \bar{x}_1 f_0(X_2) \vee x_1 f_1(X_2),$$

where $X_2 = (x_2, x_3, \ldots, x_n)$. If $f_1(X_2)$ depends on up to $(n-3)$ variables, then f can be realized by a single-rail cascade of two $(n-1)$-LUTs.

(Proof) Consider the circuit shown in Fig. 4.1, the first LUT realizes $f_0(X_2)$, while the second LUT realizes $f_1(X_2)$ and the multiplexer, which is denoted by a trapezoidal symbol. When $x_1 = 0$, the multiplexer selects the upper function f_0, while when $x_1 = 1$, the multiplexer selects the lower function f_1. Thus, the circuit realizes the given $(n-1)$-variable function. □

Application of Theorem 4.2 is easy.

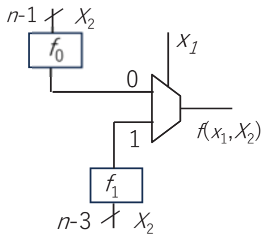

Fig. 4.1. Single-rail cascade realization of a 7-variable function using two 6-LUTs.

Theorem 4.3 *Let f be a n-variable function. Suppose that f is written as*

$$f(x_1, X_2) = \bar{x}_1 f_0(X_2) \vee x_1 f_1(X_2),$$

where $X_2 = (x_2, x_3, \ldots, x_n)$. If $f_1(X_2)$ can be represented as a product of two functions $p(X_2)$ and $q(X_2)$, and if $p(X_2)$ and $q(X_2)$ depend on at most $(n-3)$ variables each, then f can be realized by a single-rail cascade of three $(n-1)$-LUTs.

(Proof) Consider the circuit shown in Fig. 4.2. The first LUT realizes $f_0(X_2)$; the second LUT realizes $p(X_2)$ and the first multiplexer; and the third LUT realizes $q(X_2)$ and the second multiplexer and the AND gate. When $x_1 = 0$, the output is f_0, while when $x_1 = 1$, the output is $p(X_2)q(X_2)$. Thus, the circuit realizes the given n-variable function. □

In this case, $f_1(X_2) = p(X_2)q(X_2)$ can depend on up to 6 variables, and $f_1(X_2)$ has an **AND bi-decomposition**. An algorithm to detect such decompositions is available.

V. Experimental Results

A. 7 and 8 Variable Functions

Theorem 5.1 *Any 7-variable function can be realized by a single-rail cascade using five 6-LUTs.*

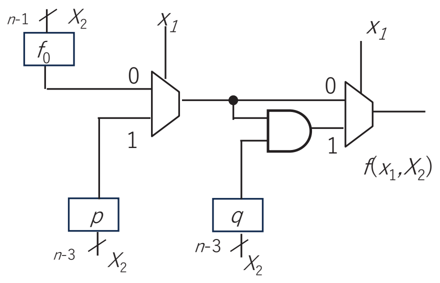

Fig. 4.2. Single-rail cascade realization of a 7-variable function using three 6-LUTs.

(Proof) Let $f(X_1, X_2)$ be a 7-variable function, where $X_1 = (x_1, x_2)$ and $X_2 = (x_3, x_4, \ldots, x_7)$.

Consider the circuit shown Fig. 5.1, where g_1 is a 6-variable function such that $g_1 = \bar{x}_2 h_0(X_2) \oplus x_2 h_1(X_2)$, while $g_2(X_2)$ and $g_3(X_2)$ are 5-variable functions.

From the circuit, we have the following relations:
When $X_1 = (0,0)$, $f(0,0,X_2) = h_0(X_2) \oplus g_2(X_2) \oplus g_3(X_2)$.
When $X_1 = (0,1)$, $f(0,1,X_2) = h_1(X_2) \oplus g_2(X_2) \oplus g_3(X_2)$.
When $X_1 = (1,0)$, $f(1,0,X_2) = g_3(X_2)$.
When $X_1 = (1,1)$, $f(1,1,X_2) = g_2(X_2) \oplus g_3(X_2)$.

From these, we have the solutions for g_1, g_2 and g_3.

$$\begin{aligned} g_1(X_2) &= \bar{x}_2 f(0,0,X_2) \oplus x_2 f(0,1,X_2) \oplus f(1,1,X_2), \\ g_2(X_2) &= f(1,0,X_2) \oplus f(1,1,X_2), \\ g_3(X_2) &= f(1,0,X_2). \end{aligned}$$

Thus, any 7-variable function can be realized by a single-rail cascade using five 6-LUTs. □

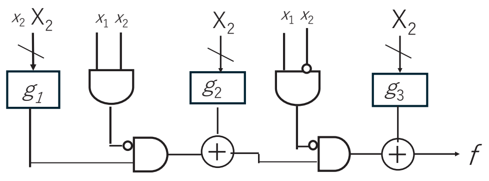

Fig. 5.1. Cascade realization for a 7-variable function.

Specifically, we have the following:

Conjecture 5.1 *Most 7-variable functions can be realized by single-rail cascades of three 6-LUTs.*

(Explanations Supporting the Conjecture) We randomly generated 10000 functions with $n = 7$ and $u = 64$ true minterms. By using **lutexact**, all the functions were realized by single-rail cascades with three 6-LUTs..

Since functions with $u = 64$ are considered as the most complicated functions, we conjecture that most 7-variable functions can be realized with three 6-LUTs. We did similar experiments for different values of u, and had the same results. Up to now, we have not encountered any 7-variable function that requires more than three 6-LUTs. □

In a similar way to Theorem 5.1, we have the following:

Theorem 5.2 *Any 8-variable function $f(X)$ can be realized by a single-rail cascade using 15 LUTs with 6-inputs.*

B. Realization using lutexact

For different pairs of (n, u), we generated 100 random functions, and applied **lutexact** [4]. To use **lutexact**, the number of LUTs must be specified in advance [13]. The maximum computation time for each function was set to 3600 seconds. We used a computer with a Ryzen 5900HX CPU on ubuntu 24.04.2 LTS using 32 GB memory. Table 5.1 shows the numbers of 6-LUTs and the realized functions within an allocated time. Currently, **lutexact** works for functions with up to $n = 10$ variables.

For $n = 8$, significant difference exist on the numbers of LUTs derived by Theorem 5.2 and experimental results. Theorem 5.2 shows that 15 LUTs are sufficient, while experimental results show that 8 LUTs are sufficient for most functions.

TABLE 5.1
FRACTION OF FUNCTIONS REALIZED BY SINGLE-RAIL CASCADES.

n	u	6-LUT	Functions
8	30	4	100
8	40	4	87
8	50	5	100
8	60	6	*99
8	70	7	100
8	128	8	100
9	20	5	100
9	25	5	100
10	4	4	100
10	15	6	100
10	20	6	*98

n: the number of input variables.
u: the number of true minterms.
*: For the remaining functions, different SAT solver found solutions in short time.

VI. CONCLUSION

This paper presented design methods for single-rail LUT cascades. They are useful to design FPGAs with *cascade multiplexer paths* among LUTs. The main results are:

1. Any $2m$-variable function can be realized by a single-rail cascade with $(m + 1)$-LUTs. The number of LUTs is at most $2^{m+1} - 1$.
2. When a $2m$-variable function has a functional decomposition $f(X_1, X_2)$, where X_1 and X_2 have m variables, and the column multiplicity of the decomposition is μ, the number of LUTs can be reduced to $2\mu - 1$.

ACKNOWLEDGMENTS

This work was supported in part by JSPS KAKENHI Grant Number JP25K15045. The author thanks to Dr. Alan Mishchenko who provided the author the latest version of **lutexact**, and important comments.

REFERENCES

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

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

[3] A. Mishchenko and T. Sasao, "Encoding of Boolean functions and its application to LUT cascade synthesis," *IWLS-2002*, pp. 115-120, New Orleans, June 5, 2002.

[4] A. Mishchenko, "ABC: A system for sequential synthesis and verification," Available online: http://www.eecs.berkeley.edu/ alanmi/abc/ (accessed on July 21, 2025).

[5] K. K. Maitra, "Cascade switching networks of two-input flexible cells," *IRE Trans. Electron. Comput.*, EC-11, pp. 136-143, 1962.

[6] A. Mukhopadhay and H. S. Stone, "Cellar logic,"in A. Mukhopadhyay (ed.), *Recent Developments in Switching Theory*, Academic Press, New York, 1971.

[7] H. Nakahara and T. Sasao, "A PC-based logic simulator using a look-up table cascade emulator," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E89-A, No.12, Dec. 2006, pp. 3471-3481.

[8] K. Nakamura, et.al, "A memory-based programmable logic device using look-up table cascade with synchronous static random access memories," *Japanese Journal of Applied Physics*, Vol. 45, No. 4B, 2006, April, 2006, pp. 3295-3300.

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

[10] T. Sasao, "Design methods for multi-rail cascades," *5-th International Workshop on Boolean Problems*, Sept. 20, 2002, Freiberg. (invited talk).

[11] T. Sasao and A. Mishchenko,"LUTMIN: FPGA logic synthesis with MUX-based and cascade realizations," *IWLS-2009*, Berkeley, CA, U.S.A., July 31-Aug. 2, 2009, pp.310-316.

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

[13] T. Sasao, "Statistical method to estimate the number of LUTs to realize sparse logic functions," *International Workshop on Logic and Synthesis*, Verona, Italy, June 13, 2025.

[14] H. S. Stone and A. J. Korenjak, "Canonical form and synthesis of cellular cascades," in *IEEE Transactions on Electronic Computers*, vol. EC-14, no. 6, pp. 852-862, Dec. 1965.

[15] A. Suzuki, S. Noguchi, and J. Oizumi, "The capability of Boolean functions realization by the $(m + 1, 1)$-type multi-stage logical networks," (in Japanese), *IEICE*, Vol. 52-C, No.8, pp, 459-486, March 1969.

[16] AMD, *Versal ACAP Configurable Logic Block Architecture Manual (AM005),* https://docs.amd.com/r/en-US/am005-versal-clb/Look-Up-Table (accessed on July 21, 2025).