Evaluating logic functionality of cascaded fracturable LUTs

GUO Zhenhong¹,², LIN Yu¹, LI Tianyi³, JIA Rui¹,², GAO Tongqiang³, YANG Haigang³

(a.Institute of Electronics; b.University of Chinese Academy of Sciences, Chinese Academy of Sciences, Beijing 100190, China)

Abstract: Look Up Tables(LUTs) are the key components of Field-Programmable Gate Arrays(FPGAs). Many LUT architectures have been studied; nevertheless, it is difficult to quantificationally evaluate an LUT based architecture. Traditionally, dedicated efforts on specific modifications to the technology mapping tools are required for LUT architecture evaluation. A more feasible evaluation method for logic functionality is strongly required for the design of LUT architecture. In this paper, a mathematical method for logic functionality calculation is proposed and conventional and fracturable LUT architectures are analyzed. Furthermore, a cascaded fracturable LUT architecture is presented, which achieves twice logic functionality compared with the conventional LUTs and fracturable LUTs.

Key words: Field Programmable Gate Array; cascaded Look Up Tables; functionality evaluation

LUT based FPGAs have been one of the most popular digital logic implementation media in various application domains over last decade. However, the search of new FPGA architecture remains a continuing area of research[1]. The crucial goal of these researches is the reduction of the delay, area, and power consumption, and improvement of resource utilization, routability and logic functionality of the programmable logic.

Reducing delay quantificationally is especially important as it facilitates high frequency implementation on FPGAs. In modern FPGAs, the delay is dominated by interconnects[2-3]. Modern FPGAs fracture a LUT into several sub-LUTs in order to achieve more flexible inputs and outputs, and improve logic functionality[4-6], however, later discussion will show that more inputs cause an exponential increase in LUT area. Another method is to add inter-LUT connections, where a LUT’s output connects to another LUT’s input via a dedicated interconnect, to avoid going through the large multiplexer before each LUT and the channels outside the logic cluster[2].

Combining the advantages of fracturable and cascaded LUTs, a new cascaded fracturable LUT architecture is proposed. The proposed LUT works as a fracturable LUT in fracturable mode. While in cascaded mode, the fractured sub-LUTs are cascaded, in order to implement different logic functionality. An SRAM controlled multiplexer is used to switch between the two modes, and a feedback path is added for cascading the sub-LUTs, which will have a negligible cost in delay and area.

Traditionally, dedicated efforts on specific modifications to the technology mapping and packing tools are required for LUT architecture evaluation[2,7-8]. In this paper, a mathematical method for logic functionality calculation is proposed. Our evaluation method shows that the proposed architecture implements twice logic functionality of conventional LUTs and fracturable LUTs.

1 Conventional LUT

In this section, a new method is employed to evaluate logic functionality of convention LUT. The definition of logic functionality and minterm used in our evaluation method refers to [9-10].

In Fig.1, conventional K-input LUT has K inputs and one output, which are represented by \(x_i, i=1,2,\ldots,K\), and \(y\) respectively. The LUT includes a \(2^K\)-input multiplexer(\(2^K\)-MUX). The \(K\) inputs act as control signals of the \(2^K\)-MUX. The inputs of the \(2^K\)-MUX are provided by
SRAM cells. So, the logic function of output \( y \), \( F(y) \), can be formulated as following:

\[
F(y) = a_0 \bar{x}_1 \cdots \bar{x}_K + a_1 x_1 \cdots x_K + a_2 x_2 \cdots x_{K-1} \bar{x}_K + \cdots + a_{K-1} x_1 x_2 \cdots x_{K-1}
\]

where \( a_i, i = 0,1,\ldots, 2^K - 1 \), are SRAM configuration bits.

\[ |F(y)| \] is denoted to be the number of logic functions of \( F(y) \). Because there are \( 2^K \) configuration bits, the number of logic functions that conventional LUT can realize is \( 2^{2^K} \) according to Theorem 1. So the below equation is obtained:

\[
|F(y)| = 2^{2^K}
\]

As the number of inputs, \( K \), increases, the number of SRAM cells will increase exponentially, leading to a huge increase in area. Meanwhile, in practical applications, mapping logic circuits to a LUT with large number of inputs will cause lots of SRAM resources wasted.

## 2  Fracturable LUT

In this section, the proposed method is extended to evaluate logic functionality of fracturable LUT.

In order to increase the number of inputs and outputs, a \( K \)-LUT can be fractured into two \((K-1)\) input sub-LUTs. The inputs of fracturable LUT are increased from \( K \) to \( 2K-1 \), and the output number is increased by 1. Each sub-LUT has \( 2^{K-1} \) SRAM cells, and total SRAM resources remain the same.

When inputs of fracturable LUT are less than \( 2K \), some inputs are shared by the two sub-LUTs. The range of the number of shared inputs, marked as \( I \), is from 0 to \( K - 1 \). As in Fig.2, \( x_i, i=1,2,\ldots,I \), are shared inputs, and \( x^j_i, i=I+1,\ldots,K-1, j=0,1 \), are exclusive inputs. Each of the two sub-LUTs has its own output \( y^0 \) and \( y^1 \).

When the fracturable LUT is used as two \((K-1)\)-input sub-LUTs, each of them can implement \( 2^{K-1} \) logic functions. The logic functions of output \( y^j, j=0,1 \), are represented as following:

\[
F(y^j) = \bar{x}_1 \cdots \bar{x}_I x^j_1 \cdots x^j_I + \cdots + \bar{x}_1 \cdots \bar{x}_{I-1} x^j_I x^j_{I+1} \cdots x^j_{K-1}
\]

\[
\bar{x}_1 \cdots \bar{x}_I x^j_I \left( a^j_{2^{K-1}-I+1} x^j_{I+1} \cdots x^j_{K-1} + \cdots + a^j_{I} x^j_{I+1} \cdots x^j_{K-1} \right) + \cdots + \bar{x}_1 \cdots \bar{x}_{I-1} x^j_{I-1} x^j_I \left( a^j_{2^{K-1}-I+1} x^j_{I+1} \cdots x^j_{K-1} + \cdots + a^j_{I} x^j_{I+1} \cdots x^j_{K-1} \right)
\]

(3)

**Theorem 1:** Two sub-LUTs with \( I \) shared inputs implement \( 2^{K-I} \) same logic functions.

From **Theorem 1**, the total number of same logic function of \( F(y^0) \) and \( F(y^1) \), represented by \( |F(y^0) \cap F(y^1)| \), is \( 2^{2^K-I} \). So,

\[
|F(y^0)| = |F(y^1)| = 2^{2^K-I}, |F(y^0) \cap F(y^1)| = 2^{2^K-I}
\]

(4)

Furthermore, the two sub-LUTs can be merged by a 2-input multiplexer to generate output \( y \). The number of logic functions that output \( y \) can realize is \( 2^{2^K} \).

\[
|F(y)| = 2^{2^K}
\]

(5)

Therefore, no matter how many inputs, the number of logic functions of output \( y \) does not change.

For output \( y \) of fracturable LUT, the number of logic functions is equal to the output of conventional LUT. However, the logic functions of the two LUT architectures are different. Conventional LUT can implement all combinational logic functions of the \( K \) inputs, however, fracturable LUT can only implement part of logic functions of all inputs.
3 Cascaded Fracturable LUT

Although fracturable LUT has more inputs and outputs compared to conventional LUT, the two architectures can implement same number of logic functions. In this section, a cascaded fracturable LUT architecture is proposed and its architecture is shown in Fig.3.

Compared with fracturable LUT in Fig.2, an SRAM-controlled multiplexer and a feedback path from output of LUT0 to input of LUT1 are added. By configuring the multiplexer, the proposed architecture works in two different modes, fracturable mode and cascaded mode. In fracturable mode, input \( x_1 \) of LUT1 is used and the feedback path is not activated, the proposed architecture acts as same as a fracturable LUT. In cascaded mode, output \( y_0 \) of LUT \( y_1 \) is used as one input to LUT1, and a LUT chain is constructed. In the following, the logic functionality of cascaded mode and fracturable mode of proposed architecture is analyzed.

3.1 Cascaded Mode

In cascaded mode, the total number of external inputs is \( 2K-2 \) as in Fig.4, where \( I=0 \) meaning no shared inputs. LUT0 has \( K-1 \) external inputs, represented by \( x^0_i \), \( i=1,2,\cdots,K-1 \), while LUT1 has \( K-2 \) external inputs, \( x^1_i \), \( i=2,3,\cdots,K-2 \). And \( y^0 \) and \( y^1 \) represent the output LUT0 and LUT1 respectively, and \( y^0 \) is connected to the input of LUT1. \( s \) acts as the selection signal of the multiplexer. The logic functions of output \( y^1 \) and \( y^0 \) can be formulated as following:

\[
F(y^1) = F(y^0)F(y^1) + F(y^0)F^{n}(y^1) \tag{7}
\]

\[
F(y) = sF(y^0)F(y^1) + sF(y^0)F^{n}(y^1) \tag{8}
\]

where

\[
F(y^0) = \sum_{x_{K-1}^0} \alpha_0^0 x_{K-1}^0 \cdots x_{1}^0 + \cdots + \alpha_0^0 x_{1}^0 \cdots x_{K-1}^0
\]

\[
F(y^1) = \sum_{x_{K-1}^1} \alpha_1^1 x_{K-1}^1 + \cdots + \alpha_1^1 x_{1}^1 \cdots x_{K-1}^1
\]

\[
F^{n}(y^1) = \sum_{x_{K-1}^1} \alpha_1^1 x_{K-1}^1 + \cdots + \alpha_1^1 x_{1}^1 \cdots x_{K-1}^1
\]

Theorem 2: The logic functions \( y^1 \) and \( y^0 \) are

\[
F(y^1) = F(y^0)F(y^1) + F(y^0)F^{n}(y^1)
\]

\[
F(y) = sF(y^0)F(y^1) + sF(y^0)F^{n}(y^1)
\]

are the sum of products. \( F(y^0) \) has \( N_0 \) configuration coefficients and \( N_0 \) complete minterms. Each of \( F^i(y^1) \) and \( F^{n}(y^1) \) has \( N_1 \) configuration coefficients. \( F^i(y^1) \) and \( F^{n}(y^1) \) have same minterms but different configuration coefficients. In each of, \( F(y^0) \), \( F(y^1) \) and \( F^{n}(y^1) \), only one product is 1 for any combination of its corresponding configuration coefficients. \( s \) denotes an input signal. Then the number of logic functions of \( y^1 \) and \( y \) are

\[
|F(y^1)| = 2^{N_1} + (2^{N_0} - 2^N)(2^{N_0} - 2)/2
\]

\[
|F(y)| = 2^N x 2 + 2^{N_0} (2^{N_0} - 2)
\]
In Equation 7 and Equation 8, \( F\left(y^0\right) \), \( F'\left(y^1\right) \) and \( F^m\left(y^1\right) \) have \( 2^{K-1} \), \( 2^{K-2} \) and \( 2^{K-2} \) configuration coefficients, respectively. From Theorem 2, the number of logic functions of \( y^i \) and \( y \) are

\[
\begin{align*}
|F(y^i)| & = 2^{K-2} + \left(2^{K-2} - 2^{K-3}\right) \left(2^{K-1} - 2\right) / 2 \approx 2^{K-1} \\
|F(y)| & = 2^{K-2} \times 2 + 2^{K-1} \left(2^{K-1} - 2\right) \approx 2^{K-1}
\end{align*}
\]

In cascaded mode, the number of inputs ranges from \( K \) to \( 2K-2 \). When inputs are less than \( 2K-2 \), as in Fig. 4, the two sub-LUTs share \( I \) inputs, \( x_i \), \( i = 2, 3, \ldots, I+1 \). LUT0 has exclusive inputs represented by \( x_i^0 \), \( i = 1, I + 2, \ldots, K-1 \), and \( x_i^0 \), \( i = I + 2, \ldots, K-1 \), are exclusive inputs to LUT1. The logic function of output \( y^i \) and \( y \) can be formulated as same as Equation 7 and Equation 8, but \( F\left(y^i\right) \) and \( F(y) \) are divided into \( 2^I \) parts according to different minterms of shared inputs. From Theorem 2, the following is obtained:

\[
\begin{align*}
|F(y^i)| & \approx \left(2^{K-I-1}\right)^2 = 2^{K-I} - 2^I \\
|F(y)| & \approx \left(2^{K-I}\right)^2 = 2^{2K}
\end{align*}
\]

### 3.2 Total logic functionality

In this subsection, the total logic functionalities of output \( y \) of the proposed architecture in both fracturable mode and cascaded mode are analyzed. \( F_F_y \) and \( F_C(y) \) are used to represent the logic functions of output \( y \) in fracturable mode and cascaded mode, respectively.

From Equation 5 and Equation 12, the below equations are obtained:

\[
|F_F(y)| = 2^{K-}\left|F_C(y)\right| \approx 2^{K}
\]

However, the total number of logic functions of \( y \) is not the summation of \( |F_F(y)| \) and \( |F_C(y)| \), because some logic functions can be implemented both in fracturable mode and cascaded mode. \( F_S(y) = F_F(y) \cap F_C(y) \) is denoted to be the same logic functions which can be implemented in the two modes, then \( |F(y)| \) which represents the total number of logic functions of output \( y \) can be computed as following:

\[
|F(y)| = |F_F(y)| + |F_C(y)| - |F_S(y)|
\]

After derivation using the proposed method, the following is obtained:

\[
\begin{align*}
|F_S(y)| & = \left(2^{K-I-1} \times 2^{K-I-2}\right)^{2^I} = 2^{K-I} \times 2^{K-I-2} \\
|F_F(y)| & \approx 2^{K-I} + 2^{K-I} - 2^{K-I} \times 2^{K-I-2} \approx 2^{K} \times 2
\end{align*}
\]

### 4 Summary and Comparisons

As a summary of previous section, the logic functionalities of conventional LUT, fracturable LUT and the proposed cascaded fracturable LUT are summarized in Table 1.

To fairly compare different LUT architectures, the same number of configurable bits, \( 2^K \) is adopted. The number of inputs to conventional LUT is \( K \). Each sub-LUT in fracturable LUT has \( K-1 \) inputs, and another input is used for output selection, so totally fracturable LUT has \( 2K-1 \) inputs. The proposed cascaded fracturable LUT also has \( 2K-1 \) inputs. In normal mode, they work as same as inputs of fracturable LUT, while in cascaded mode, only \( 2K-2 \) inputs are used...
because one sub-LUT uses the other one’s output as one of its inputs. \(I\) inputs are shared between the two sub-LUTs in fracturable LUT and the proposed architecture. Conventional LUT has 1 output, while both fracturable LUT and the proposed LUT have 3 outputs, 1 for normal mode and 2 for fracturable/cascaded mode. The number of outputs can be reduced by 1 through sharing outputs between the two modes, as in some commercial architecture.

### Table 1 Summary of logic functionalities of conventional LUT, fracturable LUT, and proposed cascaded fracturable LUT

<table>
<thead>
<tr>
<th></th>
<th>convent. LUT</th>
<th>fract. LUT</th>
<th>proposed Arch.</th>
</tr>
</thead>
<tbody>
<tr>
<td>#config. Bit</td>
<td>(2^k)</td>
<td>(2^k-1)</td>
<td>(2^k)</td>
</tr>
<tr>
<td>#input</td>
<td>(k)</td>
<td>(k)</td>
<td>2 (k)</td>
</tr>
<tr>
<td>#shared input</td>
<td>--</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>#output</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>norm. mode</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#output functionality</td>
<td>2^(k)</td>
<td>2^(k)</td>
<td>2^(k) ×2</td>
</tr>
<tr>
<td>fracturable mode</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#output functionality</td>
<td>--</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>same function between outputs</td>
<td>--</td>
<td>2^(k)</td>
<td>2^(k)</td>
</tr>
<tr>
<td>cascaded mode</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#output functionality</td>
<td>--</td>
<td>--</td>
<td>2^(k)-2</td>
</tr>
<tr>
<td>same function between outputs</td>
<td>--</td>
<td>--</td>
<td>2^(k)-2</td>
</tr>
</tbody>
</table>

Conventional LUT implements \(2^k\) logic functions. Fracturable LUT implements the same logic functionality in normal mode as conventional LUT does. In fractured mode, each sub-LUT implements half of the logic functionality. In the following, the logic functionalities of the proposed cascaded fracturable LUT are analyzed and compared.

#### 4.1 Cascaded Mode

In Fig.5, the number of logic functions of cascaded mode versus the number of inputs is presented. The logic functionality of cascaded mode is normalized to \(2^\(k\)\), the logic functionality of conventional LUT and fracturable LUT in normal mode. A family of lines is shown and each corresponds to a LUT size in terms of configuration bits. The LUT size ranges from \(2^3\) to \(2^{10}\).

Each line in Fig.5 starts from a small value, indicates that the cascaded mode implements limited logic functionality when the two sub-LUTs share their inputs. When 2 or 3 more inputs are provided, the logic functionality increases to 1 immediately, and the cascaded mode implements the same number of logic functions as conventional LUT and fracturable LUT.

![Fig.5. Logic functionality of cascaded mode](image)

#### 4.2 Same Functionality between Cascaded and Normal Node

For the proposed cascaded fracturable LUT architecture, the numbers of logic functions implemented in cascaded mode and fracturable mode are equal as in Fig.5. In Fig.6, the same logic functionality between the two modes is evaluated. The logic functionality is normalized to \(2^\(k\)\), the logic functionality of conventional LUT and fracturable LUT in normal mode. A family of lines of different LUT sizes is shown.

As in Fig.6, when the LUT is small, cascaded mode and fracturable mode have a large portion of same logic functionality.
functionality. The number of same functions is only related to the LUT size. When the LUT has more than $2^4$ configuration bits, the same logic functionality between the two modes is very close to 0 even with limited inputs.

4.3 Total Logic Functionality

The logic functionality of cascaded mode is shown in Fig.5, and the same logic functions between cascaded mode and fracturable mode are shown in Fig.6. According to Equation 16, the total logic functionality can be computed. It can be implemented by the proposed cascaded fracturable LUT. The logic functionality is normalized to $2^x$, the logic functionality of conventional LUT and fracturable LUT in normal mode.

All lines in Fig.7 representing total logic functionalities for different LUT sizes start from 1 to 1.31. Even with the two sub-LUTs sharing their inputs, the proposed cascaded fracturable LUT has the same or more logic functionality than fracturable LUT, because the cascaded mode provides more logic functionality other than fracturable mode. When 2 or 3 more inputs are provided, the total logic functionality increases immediately to $2^X$ of fracturable LUT.

![Fig.7 Total logic functionality of cascaded fracturable LUT](image)

In order to analyze the number of extra inputs required to reach the maximum logic functionality, we draw a family of lines representing total logic functionalities versus the number of exclusive inputs of each sub-LUT in Fig.8. When the number of exclusive inputs of each sub-LUT is 3, the total logic functionality reaches the maximum regardless of the LUT size.

5 Conclusion

In this paper, an evaluation method for logic functionality of LUT architectures is presented. Using this method, the logic functionalities of conventional LUTs and fracturable LUTs are evaluated. In order to improve the logic functionality of LUTs, a cascaded fracturable LUT architecture is proposed, and the logic functionality of this architecture is evaluated using the proposed method. Results show that the proposed architecture achieves twice logic functionality of conventional LUTs and fracturable LUTs.

References:


Biography:

GUO Zhenhong (1989–), female, born in Xiangyang, Hubei Province, Master’s degree. Main research field includes digital IP. email: guozh827@126.com.

LI Tianyi (1989–), male, born in Changchun, Jilin Proving, Master’s degree. Main research field includes mixed-signal circuit, especially design of DDR circuit.

GAO Tongqiang (1970–), male, born in Shijiazhang, Hebei Province, associate professor. Main research field includes CMOS radio-frequency/analog integrated circuit.

Liu Yu (1982–), male, born in Zhuhai, Guangdong Province, Master’s degree. Main research field includes architecture of FPGA, FPGA CAD tools, high-level synthesis of FPGA and high performance computing, etc.

JIA Rui (1988–), male, born in Taiyuan, Shanxi Province, Doctor’s degree. Main research field includes reconfigurable computing.

YANG Haigang (1960–), male, born in Shanghai, professor. Main research field includes mixed-signal integrated circuit and very large scale integrated circuit.

第三届燃烧流场的光学诊断技术学术研讨会

燃烧过程复杂多变，对瞬态环境的实时诊断技术要求极其苛刻。燃烧流场的光学诊断技术主要是以激光技术、光电技术、数据处理技术等为基础的一种综合性测试诊断技术，可以实现燃烧场温度、组分及浓度、火焰构造和流速等参量信息的实时精确测量，而且测量对燃烧过程无扰动。这些参数的测量对于研究燃烧场的燃烧化学反应动力学过程，如固体推进剂燃烧动力学、超声速燃烧动力学、汽车和飞机发动机燃烧效率和污染控制、以及保障电站锅炉安全和经济运行等具有重要意义。

为了促进我国本领域技术的完善与发展，学会定于2016年11月在西安召开“第三届燃烧流场的光学诊断技术学术研讨会”，会议组委会将邀请国内外该领域的知名专家和学者到会共同交流，深入探讨燃烧流场的光学诊断技术领域所取得的最新研究成果。诚挚欢迎国内外相关领域研究院所的科研人员以及高等院校的教师、研究生踊跃参加。

主办单位：中国工程院信息与电子工程学部
国家自然科学基金委员会
中国光学工程学会

承办单位：中国光学工程学会

中国宇航学会光电专委会

联办单位：西北核技术研究所，华南理工大学

超声速冲压发动机技术重点实验室（绵阳分部）

大会主席：乐嘉陵 院士（中国空气动力研究与发展中心）

会议形式：大会报告、专题报告、Poster、专题讨论、展览展示等

中国光学工程学会

联系人：吴迪，022-58168520, wudi@cnoenet.com; 蔡方方，022-85168541, cai_ff@csoe.org.cn

征文范围（详见会议网站：http://www.csoe.org.cn/cd2016/）

专题一：燃烧诊断技术

专题二：复杂燃烧流场特性测量

专题三：先进发动机点火技术

包括以上方面，但不局限于此。欢迎与空间光通信有关的所有来稿。

投稿须知：会议邀请作者将撰写的论文投稿至本会议。文章长度为4~8页，中英文兼收，所有文章必须严格符合会议征稿主题，投稿论文必须是从未在任何会议、期刊及杂志上出版。投稿请登录在线投稿系统 http://events.kjtw.com/cougo/1426492999.html

论文发表：会议期间将评选在会议论文集中。其中，中文优秀稿件推荐至《红外与激光工程》EI、《光学精密工程》EI、《强激光与粒子束》EI、《实验流体力学》中英文核心、《太赫兹科学与电子信息学报》科技核心，正刊出版；英文稿件推荐至SPIE会议论文集，EI核心检索。

全文截稿时间：2016年7月30日