Reduced Look Up Tables With Increased Functionality
Keith H. Gudger (408) 476-8938
P.O. Box 336
Soquel CA 95073 USA

Abstract

This paper proposes a method to reduce the size of 'n' input FPGA look up tables (LUTs) to less than 2^n bits. The reduced LUT (RLUT) requires more inputs, which also increases the functionality of the LUT. The increase in the number of inputs is a trade-off against the smaller size of the RLUT. RLUTs for n = 4 should decrease total chip area while increasing logic capacity. The RLUT performance is equal to or better than standard LUTs.

Introduction

Commercial FPGAs mostly fall into one of two categories: look up table (LUT) based [1-4,7,8] or multiplexer based [5,6]. The LUT devices use SRAM technology while the multiplexer based devices use anti-fuses. One key difference is that the FPGAs with multiplexer cells only have to program the routing to implement various logic functions. LUT based devices use SRAM bits to connect signals to LUTs composed of separate SRAM bits.

The multiplexer based devices trace their origins to research on universal logic modules (ULMs) [9-15]. In [15] Chen and Hurst suggested the multiplexer as the best choice for a ULM.2 (2 inputs) (figure 1a). The ULM used in the early Actel devices [5] is similar. A ULM.n can implement all functions of 'n' variables, but requires 'M' inputs. Note that 'M' is larger than 'n'. For example, a ULM.2 requires 3 connections to implement any function of 2 variables (figure 1a). One input, labeled x1, is one of the two primary signals. The other two inputs are of the set {0, 1, x0, !x0}.

Many logic functions require more than 2 inputs. One way to form ULMs for larger numbers of inputs is to concatenate ULMs. A ULM.3 built in this manner has 6 inputs (figure 1b). Two papers previously proposed ULM.3s with only 5 inputs [12,14]. Figure 1c shows one version of this type of ULM.3 [15].

"It is natural and cost-effective to use PROMs to realize arbitrary functions." [19] This is why many FPGAs use look-up tables to implement logic. LUTs consist of two parts: the memory itself and an input decoder. The memory size is 2^n, where 'n' is the number of inputs. A LUT can form any logic function of the 'n' inputs. FPGAs generally use LUTs with 'n' varying from three [7] to five [2,3]. However, as demonstrated in [16,17], LUTs with 4 inputs are the most efficient across various technologies. A number of commercial FPGAs use LUTs with 4 inputs [1,4,8]. The device in [4] combines two 4-input LUTs with a 3-input LUT to provide functions of from 5 to 9 variables.

K.H. Gudger, Reduced LUTs With Increased Functionality, Page 1
In [11], Edwards stated that ULMs could provide "memory by connection." He further stated that "in a ... machine employing such devices it would be impossible to distinguish between hardware employed for 'memory' and that employed for 'functional computation'." This leads us to using a ULM.x to implement the memory required for the n input LUTs.

The Reduced LUT (RLUT)

Figure 2a shows a schematic of a tree-decoded LUT of 3 inputs. Note that this LUT is a combination of 2 input multiplexors and is similar to figure 1b. The portion in the dashed box is the same as a multiplexor ULM.3. Two inputs drive a 1 of 4 decoder, and four inputs are from the set \(0,1, x0, !x0\). Eight SRAM bits provide this input set using another level of multiplexors.

Since the 'core' of the LUT is a ULM.3, this leads to the question of whether the optimized ULM.3 of figure 1c can replace this core. Doing so leads to the reduced LUT of figure 2b. Its core circuit is the same as figure 1c. For the case of the 3 input LUT, this circuit reduces the number of required memory elements from 8 to 6, a 25% reduction. However, the number of inputs may increase from 3 to 6.

Even higher percentage reductions of the memory required are possible for larger LUTs. Table 1 shows key data points for LUTs built using this approach.

<table>
<thead>
<tr>
<th>&quot;n&quot;</th>
<th>number of bits:</th>
<th>percentage reduction</th>
<th>Minimum inputs:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LUT</td>
<td>RLUT</td>
<td>ULM.n</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>18</td>
<td>13</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>34</td>
<td>21</td>
</tr>
<tr>
<td>7</td>
<td>128</td>
<td>54</td>
<td>33</td>
</tr>
<tr>
<td>...</td>
<td>2^n</td>
<td>2*(n-1)/2</td>
<td></td>
</tr>
<tr>
<td>odd n</td>
<td>2^n</td>
<td></td>
<td></td>
</tr>
<tr>
<td>even n</td>
<td>2^n</td>
<td>2*(n-1)+2</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3a and 3b show RLUTs for 4 and 5 inputs, respectively. The derivation of the RLUT for 4 inputs follows from Chen and Wu's work in [14]. This example uses multiplexors and requires 8 inputs. A bit reduction of 37% requires a 100% increase in the number of inputs. One way to make a RLUT for 5 inputs is to concatenate 3 input RLUTs. This structure requires 13 inputs, which is a 160% increase in input terminals for a 44% decrease in bits.
So far, we are losing ground in our quest to reduce the effective area consumed by look-up tables. We need to find a way to reduce the number of inputs. First consider the 3 input LUT case. In [14] Chen and Wu grouped the 256 possible min-terms for ULM.3s into 16 canonical groups. For 15 of these 16 terms, inputs \( \{a,b,c\} \) can all be tied to \( x0 \), bringing the number of inputs back to 3 for this 3 input LUT. However, to cover 1 term, \( f_p(x1x2 + !x1x0) \), either input \( a \) or \( c \) must be separate from \( b \). This means that we can connect input \( b \) to input \( c \). This gives us 4 inputs for the 3 input LUT, \( \{x1,x2,a,b\} \). This ULM now has 33% more input signals for a 25% reduction of RAM bits. This is still not a particularly good trade off.

Next consider the 4 input LUT. We currently have a 8 input set \( \{x1,x2,x3,a,b,c,d,e\} \). In [10] Preparata and Muller suggest that the lower bound for inputs for a ULM.4 is 5. By inspection of the required min-terms, we can connect inputs \( a, c, \) and \( d \) together, and inputs \( b \) and \( e \) together. This reduces the input set to 5, \( \{x1,x2,x3,a,b\} \). In this case there is a 25% increase in the number of required inputs for a 37% decrease in the number of bits. This circuit reduces total chip area if the reduction in bit area is more than the area required for the extra inputs.

In the case of the 5 input LUT, there are 13 inputs to start with. Theory [10] suggests that we could reduce this to 9 inputs. However, this leaves us with a 80% increase in required inputs for a 44% decrease in LUT size. The numbers get worse as the LUT size increases.

Figure 4 summarizes this discussion.

Routing Consequences using the RLUT

The 4 input RLUT is the most promising. The proposed circuit requires 5 inputs instead of 4 and has 10 memory bits instead of 16. Since each cell has at least one output, the actual number of cell connections is 6. The standard LUT requires at least 5. In the case of [4], there are 16 signals connected to the logic block. Replacing the two 4-input LUTs with the RLUT increases the number of connections to 18, a 12.5% increase.

The first consideration is whether the routing channels feeding the LUT cells have to increase. In [17] Rose et al. addressed whether increasing the number of inputs to a LUT increases the size of the routing channel. Their results show that the maximum channel width does not need to change as 'n' increases from 4 to 8. They also calculate El Gamal's [18] expected value of the number of tracks. This value only increases by 0.2 tracks as inputs increase from 4 to 6. This leads to the conclusion that the increased input requirements of the RLUT need not increase the width of the routing channels.
However, each added input requires a set of switches from the channel to that input. In the case of the 4-input RLUT, which has six fewer bits than a standard LUT, we are ahead in terms of area as long as the extra input requires fewer than six switches. (This assumes that input switches are the same size as LUT bits. In practice, switch bits are slightly smaller.)

The actual number of inputs: \( n \) effective

Since the RLUT has 5 inputs instead of 4, it can implement some functions of 5 inputs and all functions of 4 inputs. This means that the RLUT logic block, besides having a smaller bit area than a standard LUT, can implement more logic functions. The RLUT has an \( 'n' \) effective between 4 and 5.

In [16], Kouloheris and El Gamal state that the ratio of memory bit area to fixed area in the logic block largely determines the best \( 'n' \) for total chip area. They proposed the following formula for calculating the optimum \( 'n' \):

\[
(1.17 \log_2 n - 0.17) \times (\text{Area(Bits)} + \text{Area(Fixed)})
\]

Since the RLUT decreases the memory bit area, it increases the optimum \( 'n' \). For the RLUT with \( n = 4 \), we can multiply the area of the bits by 10/16. For a fixed area 20 to 30 times the bit area, the optimum \( 'n' \) is between 4 and 5. This is in the range of the \( 'n' \) effective of the RLUT with \( n = 4 \).

The RLUT with \( n = 4 \) can actually increase the logic capacity for a given chip area. This is true even if the increase in required routing bit area equals the decrease in logic block area. In the above equation, the left hand side decreases with increasing \( 'n' \), while the right hand side increases. In the case of an RLUT with an \( 'n' \) effective larger than \( 'n' \), the bit area is fixed, while the left side \( 'n' \) is \( 'n \) effective'. As \( 'n \) effective' increases, the logic capacity of the device increases. Table 2 shows the increase in logic capacity as \( 'n \) effective' increases from 4.0 to 4.5.

<table>
<thead>
<tr>
<th>( 'n ) effective</th>
<th>normalized chip area</th>
<th>% improvement in logic capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>4.1</td>
<td>0.975</td>
<td>+ 2.6%</td>
</tr>
<tr>
<td>4.2</td>
<td>0.952</td>
<td>+ 5.0%</td>
</tr>
<tr>
<td>4.3</td>
<td>0.929</td>
<td>+ 7.6%</td>
</tr>
<tr>
<td>4.4</td>
<td>0.91</td>
<td>+ 9.9%</td>
</tr>
<tr>
<td>4.5</td>
<td>0.89</td>
<td>+12.4%</td>
</tr>
</tbody>
</table>

If \( 'n \) effective' is 4.4 for the RLUT, then we can expect a 10\% improvement in logic capacity.
Performance Impact

There are several ways to implement a 4-input LUT. Two possible ways are tree decoders (figure 2a) or decoded subsets of the input pins. SPICE analysis of the circuit in figure 5a on a 0.8 micron $l_{eff}$ process shows the circuit is 10% to 16% faster than a pure tree or an unequal partially decoded scheme. This circuit uses two identical 2:4 decoders. Each decoded output drives four sets of 1 of 4 select transistors. Between each memory cell and the output are two series transistors.

To modify this circuit for the RLUT, one must change only the first decoder. Instead of the usual four 2-input NOR gates each driving four transistors, we now have six 2-input gates driving one or two transistors. While there are more gates, the drive requirements on each are at least cut in half. This means that each driving gate can be approximately one-half of its previous size for the same performance. Each NOR gate input previously drove two NOR gates with width = W. Each decoder input now drives two gates with width = W/2. In the worst case, each decoder input now drives as much gate oxide capacitance as before. There is approximately a 75% decrease in total gate oxide capacitance, which means that the RLUT circuit will be faster than the LUT. In a SPICE simulation, the RLUT with one-half sized input drivers was the same speed as a standard LUT. This means that three out of five of the RLUT inputs have reduced input capacitance. This translates to increased performance of the device.

Conclusion

This paper presents a method to reduce the bit count of look-up table based FPGAs. The reduced LUT, which requires more input connections, can implement more logic functions than the standard LUT. Even taking into account the increased input count, an RLUT with $n = 4$ is the same size or smaller than a standard 4-input LUT. Furthermore, its logic capacity is up to 10% more. The performance of the RLUT should be better than the standard LUT. As a next step, one could determine the impact of the RLUT on a complex logic block such as the Xilinx 4000 series. This logic block has two 4-input LUTs, one 3-input LUT, two flip-flops, many multiplexors, and sixteen inputs and four outputs. As the complexity of the logic block increases, the increase in input connections becomes less of an issue.
References


(a) is a two-to-one multiplexor ULM.2. (b) is a four-to-one multiplexor ULM.3 (six inputs). (c) is a ULM.3 (five inputs).

Figure 1

K.H. Gudger, Reduced LUTs With Increased Functionality, Page 7
Figure 2
(a) is a 3-input LUT. The dashed box is similar to figure 1b.
(b) is an RLUT with n=3. The dashed box is similar to figure 1c.
Figure 3

(a) is an RLUT with n=4 based on a ULM.4. (b) is a ULM.5.

K.H. Gudger, Reduced LUTs With Increased Functionality, Page 9
Figure 4
Percent increase in inputs and decrease in memory bits vs. n.

Figure 5
(a) Standard LUT speed path. (b) RLUT speed path.

K.H. Gudger, Reduced LUTs With Increased Functionality, Page 10