Date of publication SEP-30, 2021, date of current version JUN-07, 2021.
www.computingonline.net / computing@computingonline.net

Print ISSN 1727-6209
Online ISSN 2312-5381
DOI 10.47839/ijc.20.3.2282

A Decoder – Look up Tables for FPGAs

SERGEY F. TYURIN[1,2], RUSLAN V. VIKHOREV[3]

[1] Department of Automation and Telemechanic, Perm National Research Polytechnic University, Russia, Perm, 614990, 29 Komsomolsky prospect, (e-mail: tyurinsergfeo@yandex.ru)

[2] Department of Software Computing Systems, Perm State University, Perm, Russia, 614990, 15 Bukireva Street, (e-mail: tyurinsergfeo@rambler.ru)

[3] Perm scientific-industrial instrument making company, Perm, Russia, 614095, Karpinskogo 87, 101, (e-mail: Vihrusvla@gmail.com)

Corresponding author: Sergey F. Tyurin (e-mail: tyurinsergfeo@yandex.ru).

Great thanks to the PhD Irina A. Barinova (Perm National Research Polytechnic University). This work was supported in part by a grant from EU by TEMPUS-GreenCo (530270-TEMPUS-1-2012-1-UK- TEMPUS-JPCR) with the assistance of the Department "Computer Systems and Networks" Kharkiv National Aerospace University named after V. E. Zhukovsky "HAI".

ABSTRACT The FPGA **(**Field-Programmable Gate Array) has recently become the popular hardware and so-called LUTs (Look up Tables) are the basic of the FPGAs logic. For example, n-LUT is the MOS pass transistors multiplexer 2^n-1 which input data receive SRAM cells logic function configuration (user's projects Truth Table). Address inputs of the LUT are the variables. Therefore, we get one n-arguments logic function for the actual FPGA configuration. To get m functions (even with the same n-arguments) we should take m LUT. Authors propose a novel Decoder n-LUT (n-DC LUT), which makes possible to get m functions with the same n-arguments, like in Program Logic Array (PLA) CPLD **(**Complex Programmable Logic Device). DC LUT activates one of the 2^n product terms outputs. Combined with OR product terms we can get m functions with the same n-arguments. To do this option we can use, for example, FPGAs typical connections units. The restriction of Meade-Conway for the FPGAs allows n=3 in one tree. Two 3-LUTs with one 1-LUTs form 4-LUT. Modern Adaptive Logic Modules (ALM) have n=8, but not all possible functions are implemented. The article deals with the design and investigation of some variants 3-DC LUT: with pull up output resistors, with orthogonal output circuits, with orthogonal transistors for each pass transistor. Simulation confirms the feasibility of the proposed method and shows that DC LUT with orthogonal output circuits is better variant of the systems realization in terms of current consumption and time delay at large n. A further development of the ALM concept may be the introduction of adaptive DC LUT, which, by tuning, can calculate single LUT function or 2^n decoder functions. The proposed elements allow to increase the functionality of the FPGAs.

KEYWORDS Architecture; CMOS; FPGA Synthesis; Layout.

I. INTRODUCTION

A. MOTIVATION

LOOK UP TABLE (LUT) is a simplest, elementary FPGAs Logic Unit [1]. This logic realization started from MUX (multiplexor) and single output ROM (read only memory) universal logic modules direction, using Canonical Disjunctive Normal Form (CDNF) or Minterm Canonical Form (MCF). Another direction used PLA and PAL (Programmable Array Logic) using DNF representation of the logic functions, which later led to the CPLD creation. FPGA and CPLD are two competing areas of programmable devices. These solutions are equal and have their own strengths and weaknesses, so attempts to create hybrid devices do not stop [2, 3]. However, this direction is mainly associated with attempts to introduce PLAs into FPGAs and use them in conjunction with LUTs. This, of course, increases the bit depth of the implemented systems of functions, but, in turn, leads to the complication of the FPGA manufacturing technology. Known examples of improving FPGAs do not use the abilities of a set of variables decoding to implement systems of logical functions in FPGAs [4-6]. Therefore, the article discusses this new proposed direction of implementation of programmable logic.

B. STATE OF THE ART

Linear representation of the 1-LUT's logic function [7-9] is the following:

$$z(x,d) = d_0 \cdot \bar{x} \vee d_1 \cdot x, \qquad (1)$$

where d_0, d_1 are configurations data of the one argument (n=1) $z(x)$ function. Combining d_0, d_1 we can get 2^2 functions (Fig1).

For example, if $d_0 = 1, d_1 = 0$ we get the NOT function:

$$z(x,1,0) = 1 \cdot \bar{x} \vee 0 \cdot x = \bar{x}, \qquad (2)$$

where x – input variable.

If $d_0 = 0, d_1 = 1$ we get the x function,

$$z(x,0,1) = 0 \cdot \bar{x} \vee 1 \cdot x = x. \qquad (3)$$

Sometimes any LUTs used like connectors, so configurations data are $d_0 = 0, d_1 = 1$.

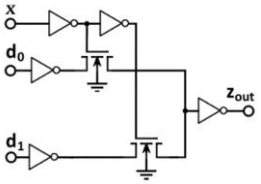


Figure 1. 1-LUT tree.

Fig. 1 shows 1-LUT according to tree representation of (1), with two MOS-p pass transistors [10],[11],[12] two configuration inputs d_0, d_1, one input variable (x) and single output function (z). NOT gates (invertors) are an amplifiers, the signal's restoration elements, one x-invertor realizes NOT(x) signal.

Linear representation of the 2-LUT's logic function is the next

$$z(x_2 x_1 d) = d_0 \cdot \bar{x}_2 \bar{x}_1 \vee d_1 \cdot \bar{x}_2 x_1 \vee d_2 \cdot x_2 \bar{x}_1 \vee d_3 \cdot x_2 x_1, \qquad (4)$$

where d_0, d_1, d_2, d_3 – configurations data of the two arguments function $z(x_2 x_1)$.

Combining d_0, d_1, d_2, d_3, we can get 2^4 functions.

For example, if $d_0 = 0, d_1 = 1, d_2 = 1, d_3 = 0$, we get XOR function.

Connecting three of the basic 1-LUT trees (without some of the NOT gates), we can design 2-LUT tree – Fig. 2.

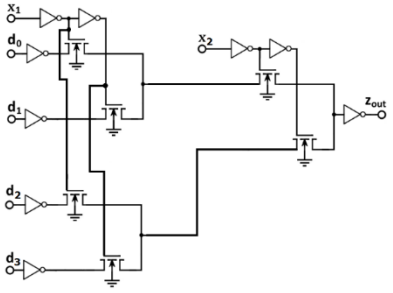


Figure 2. 2-LUT tree.

Linear representation of the 3-LUT's logic function is expression (3)

$$\begin{aligned} z(x_3 x_2 x_1 d) = {} & d_0 \cdot \bar{x}_3 \bar{x}_2 \bar{x}_1 \vee d_1 \cdot \bar{x}_3 \bar{x}_2 x_1 \vee d_2 \cdot \bar{x}_3 x_2 \bar{x}_1 \vee \\ & d_3 \cdot \bar{x}_3 x_2 x_1 \vee d_4 \cdot x_3 \bar{x}_2 \bar{x}_1 \vee d_5 \cdot x_3 \bar{x}_2 x_1 \vee \\ & \vee d_6 \cdot x_3 x_2 \bar{x}_1 \vee d_7 \cdot x_7 x_2 x_1. \end{aligned} \qquad (5)$$

Connecting two 2-LUT and one 1-LUT, we can get 3-LUT tree – Fig.3.

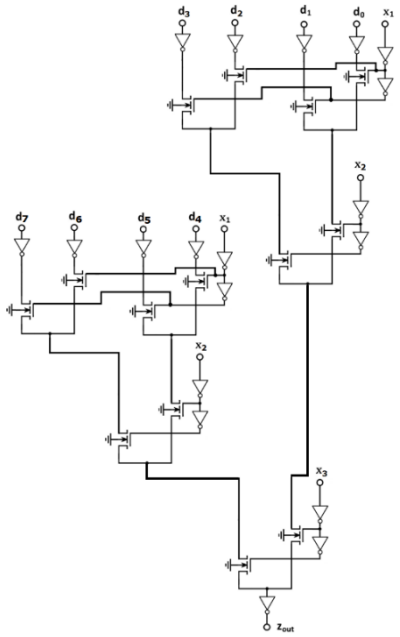


Figure 3. 3-LUT tree.

Combining $d_0, d_1, d_2, d_3, d_4, d_5, d_6, d_7$ we can get 2^8 functions.

Each branch of the tree (3) $x^{\sigma_3}{}_3 x^{\sigma_2}{}_2 x^{\sigma_1}{}_1$, where $\sigma_i \in \{0,1\}$ is indicator of the negation presence (=1) or negation absence (=0) is orthogonal to another branches. So only one branch activates [13-15].

Due to Meade-Convey restrictions [16] on the number of series-connected transistors (not more than three) 1, 2, 3-LUTs are the main FPGA's logic gates. Meade-Convey restriction [16] requires restoration after each third pass transistors link. 4-LUT and another (Adaptive Logic Modules has 5-LUT, 6-LUT and even more [11]) are created as 3-LUTs composition.

However, all n-LUTs produce only single logic function of n arguments in the canonical disjunctive normal form (CDNF) or minterm canonical form (MCF).

At the same time, each minterm can activate other logic functions of the same arguments (for example sum and carry functions). Combining this minterms by OR we can get multi-outputs logic element. CPLD, in contrast to FPGA, uses multiple output PLA technology, which uses DNF representation of the logic functions.

C. OBJECTIVES AND STRUCTURE

In the article, so-called a Decoder - Look up Tables (DC LUT) is proposed for the realization of decoding the binary vector and the multi-output logic element. Decoder (DC) is used to modify FPGAs LUT for the realization of multiple output units, based on CDNF. To solve this problem, the authors perform:

- synthesis and analysis of the proposed DC LUT by modifying known DC circuit (section 2);
- comparing the complexity in the number of transistors of the obtained solution with the known (section 3);
- layout simulations of the proposed DC LUTs and comparing the layout square, dynamic power consumption and time delay (section 4).

II. SYNTHESIS AND ANALYSIS OF THE PROPOSED DECODER LUT

Decoder or DC LUT is almost the reverse LUT, for example, 1-LUT – Fig.4.

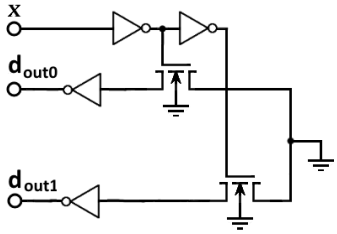


Figure 4. Reversed 1-LUT

Unlike Fig. 1, the input signal (constant) will be on the right, and the output signals will be on the left.

Linear DC 1-LUT representation is the next:

$$\begin{cases} z(\bar{x}) = d_{in} \cdot \bar{x}; \\ z(x) = d_{in} \cdot x. \end{cases} \quad (6)$$

In case x=1 input of the dout0 inverter (Fig. 4) will become disconnected to “Ground”. In case x=0 input of the dout1 invertor will became disconnected to “Ground”. Pull-up resistors usually solves the orthogonal problem in the invertor’s inputs, as shown in Fig. 5. Using additional two transistors to reverse 1-LUT (Fig.4) we get next variant of the orthogonal problem solving, 1-DC–LUT with orthogonal outputs (s0,s1) is shown in Fig. 6.

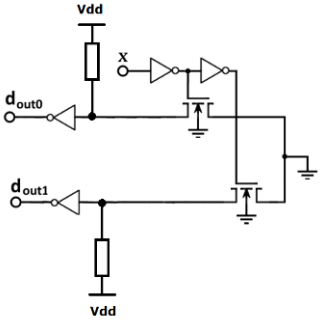


Figure 5. DC-LUT-R with pull-up resistors

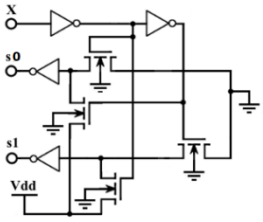


Figure 6. DC-LUT-O with proposed orthogonal outputs (s0,s1)

Additional transistors (Fig. 6) eliminate the undefined state of the inputs of inverters connected to s0 s1 without using pull-up resistors (Fig. 5).

Linear DC 2-LUT representation without orthogonal transistors is the next:

$$\begin{cases} z_0(x_2x_1) = d_{in} \cdot \bar{x}_2\bar{x}_1; \\ z_1(x_2x_1) = d_{in} \cdot \bar{x}_2x_1; \\ z_2(x_2x_1) = d_{in} \cdot x_2\bar{x}_1; \\ z_3(x_2x_1) = d_{in} \cdot x_2x_1. \end{cases} \quad (7)$$

2-DC–LUT-O with proposed orthogonal outputs [14],[15],[16] (s0,s1,s2,s3) is shown in Fig. 7.

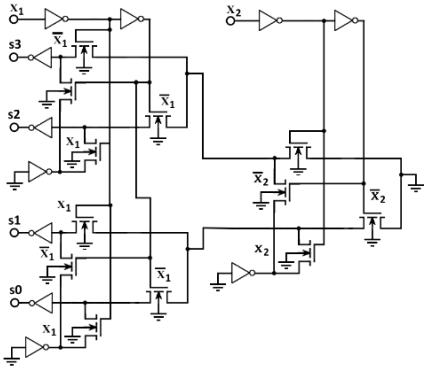


Figure 7. 2-DC-LUT-O with orthogonal outputs (s0,s1,s2,s3)

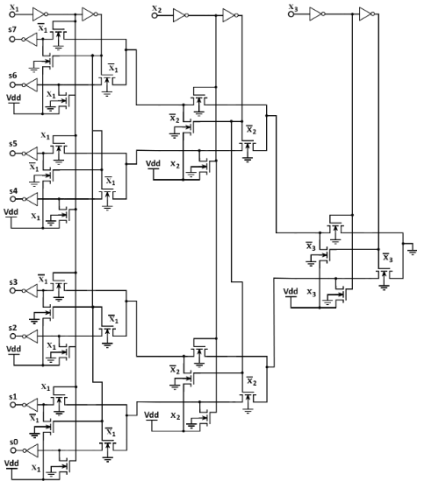


Figure 8. 3-DC-LUT-O with orthogonal by each transistor

Linear DC 3-LUT representation without orthogonal transistors is the next:

$$\begin{cases} z_0(x_3x_2x_1) = d_{in}\cdot\bar{x}_3\bar{x}_2\bar{x}_1; \\ z_1(x_3x_2x_1) = d_{in}\cdot\bar{x}_3\bar{x}_2x_1; \\ z_2(x_3x_2x_1) = d_{in}\cdot\bar{x}_3x_2\bar{x}_1; \\ z_3(x_3x_2x_1) = d_{in}\cdot\bar{x}_3x_2x_1; \\ z_4(x_3x_2x_1) = d_{in}\cdot x_3\bar{x}_2\bar{x}_1; \\ z_5(x_3x_2x_1) = d_{in}\cdot x_3\bar{x}_2x_1; \\ z_6(x_3x_2x_1) = d_{in}\cdot x_3x_2\bar{x}_1; \\ z_7(x_3x_2x_1) = d_{in}\cdot x_3x_2x_1. \end{cases} \quad (8)$$

3-DC–LUT-O with proposed orthogonal outputs (s0,s1,..,s6,s7) is shown in Fig. 8. Fig. 8. represents DC–LUT-O with orthogonalization relating to each of the pass transistors. Note, that expressions (4), (5), (6) do not take into account orthogonal problem. These are input decoding expressions, but they can be combined by OR to obtain a system of functions. For example (Fig. 7):

$$\begin{aligned} f_1(x_2x_1) &= \bar{x}_2x_1 \vee x_2\bar{x}_1 = s_1 \vee s_2, \\ f_2(x_2x_1) &= x_2x_1 = s_3. \end{aligned} \quad (9)$$

III. ANALYSIS OF LUT / DC LUT COMPLEXITY IN TRANSISTORS

The n-LUT's complexity in amount of the transistors (taking into account SRAM cells for the functions configuration, not showed in Fig.1–3) is expression (10):

$$L_{n-LUT} = (2^{n+1}-2) + 8\cdot 2^n + 4n + 2, \quad (10)$$

where $2^{n+1}-2, n=1,2,3$ – amount of the tree pass transistors; $8\cdot 2^n$ – amount of the SRAM cells transistors (6 transistors in one cell) +input data invertors transistors; $4n$ – amount of the input variables invertors transistors; 2 – amount of the output invertor transistors.

We see an exponential dependence of complexity on the number of variables. Simplifying (10), we get formula (11):

$$\begin{aligned} L_{n-LUT} &= 2\cdot 2^n + 8\cdot 2^n + 4n = 10\cdot 2^n + 4n = \\ &= 5\cdot 2^{n+1} + 4\cdot n. \end{aligned} \quad (11)$$

Expression (11) describes conditional complexity $O(2^{n+1})$ of the single logic function realization without restrictions [13]. We can design 2-LUT like (1-LUT)+(1-LUT)+(1-LUT) pay attention to restrictions [13].

Then for 3-LUT: (1-LUT +1-LUT +1-LUT)+(1-LUT +1-LUT +1-LUT) +1-LUT =3-LUT. Another variant is 2-LUT +2-LUT +1-LUT =3-LUT.

So for 4-LUT: 3-LUT +3-LUT +1-LUT =4-LUT; (2-LUT +2-LUT +1-LUT)+(2-LUT +2-LUT +1-LUT)+1-LUT=4-LUT.

Then for 5-LUT: (3-LUT +3-LUT +1-LUT)+(3-LUT +3-LUT +1-LUT)+1-LUT=5-LUT;

3-LUT +3-LUT +3-LUT +3-LUT+2-LUT =5-LUT.

To minimize trees levels (3-3 is better than 3-1-1, 2-2-2 better than 1-1-1-1-1-1) let design max decomposition by, for example, max r=3:

$$\psi_r = \left\lfloor \frac{n}{r} \right\rfloor, \left\lfloor \frac{6}{3} \right\rfloor = 2; \left\lfloor \frac{5}{3} \right\rfloor = 1; \quad (12)$$

with finite r_f:

$$\psi_{r_f} = n - r\left\lfloor \frac{n}{r} \right\rfloor; 5 - 3\cdot\left\lfloor \frac{5}{3} \right\rfloor = 2; 6 - 3\cdot\left\lfloor \frac{6}{3} \right\rfloor = 0; \quad (13)$$

Where $\lceil\ \rceil$ the round up (or take the ceiling or ceiling n/r function).

Then amount of the r-LUTs (amount of the r_f LUT always=1):

$$\psi_{r-LUT} = \sum_{i=1}^{\left\lfloor \frac{n}{r} \right\rfloor} 2^{n-ir}. \quad (14)$$

It is easy to see why n-tree complexity is

$$L_{n-LUT\_tree} = 2^{n+1} - 2. \quad (15)$$

Therefore, n-LUT scaling by max r-LUT without configuration complexity and fan-out of the input inverters gives expression (16):

$$L_{n-LUT\_tree} = 2^{n+1} - 2. \quad (16)$$

where $2\cdot\sum_{i=1}^{\left\lfloor \frac{n}{r} \right\rfloor} 2^{n-ir}$ - number of the restoration blocks (invertors) transistors, 2-number of the transistors in single r_f LUT's invertor.

Taking into account Fig.8 and expression (16) we can get n-DC–LUT-O complexity:

$$\begin{aligned} L_{n(\max r)-DC-LUT-O} &= 2\cdot(2^{n+1}-2) + 8\cdot 2^n + \\ &+ 4n + 2\cdot\sum_{i=1}^{\left\lfloor \frac{n}{r} \right\rfloor} 2^{n-ir} + 2. \end{aligned} \quad (17)$$

The authors have developed and researched several variants of the device. Second proposed variant is the block of the orthogonal additional transistors called the block of the canonical form – BCN (canonical conjunctive normal form – CCNF). 3-DC-LUT-BCN is shown in Fig. 9.

For example, minterm $\overline{x}_3\overline{x}_2\overline{x}_1$ requires orthogonal maxterm $x_3 \vee x_2 \vee x_1$ du to $\overline{\overline{x}_3\overline{x}_2\overline{x}_1} = x_3 \vee x_2 \vee x_1$. This BCNs connects to invertors inputs. Therefore, we have complexity (18):

$$L_{n(\max r)-DC-LUT-BCN} = (2^{n+1}-2)+8\cdot 2^n + n\cdot 2^n + 4n + 2\cdot\sum_{i=1}^{\left\lfloor \frac{n}{r} \right\rfloor} 2^{n-ir} + 2. \quad (18)$$

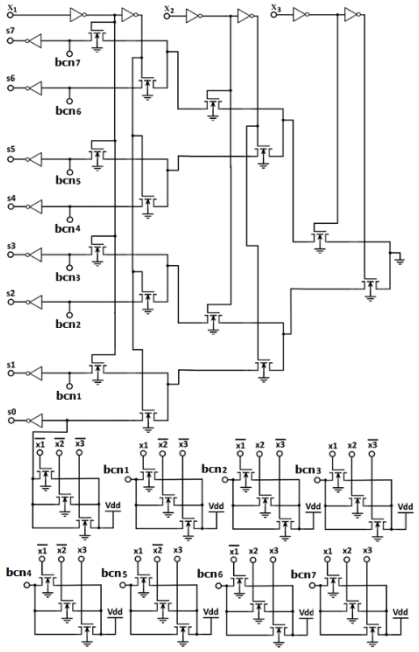


Figure 9. Proposed 3-DC-LUT-BCN with orthogonal by outputs (s0,s1,..,s6,s7)

In the input of the output invertor *s(i)* (Fig.8, Fig.9) all signals are orthogonal due to s signal is one hot code (only one is active=1). To calculate m function it needs H signals are the configuration information, *H(j)*=1 if the *i*-function (si) include *j*-maxterm. Single disjunctive block for n arguments showed on Fig.10.

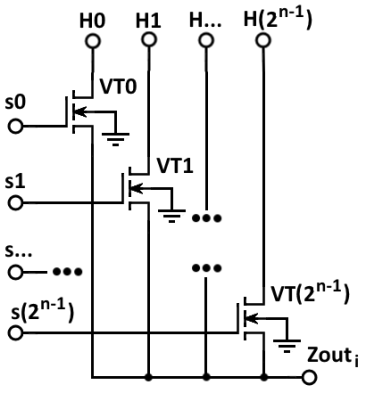
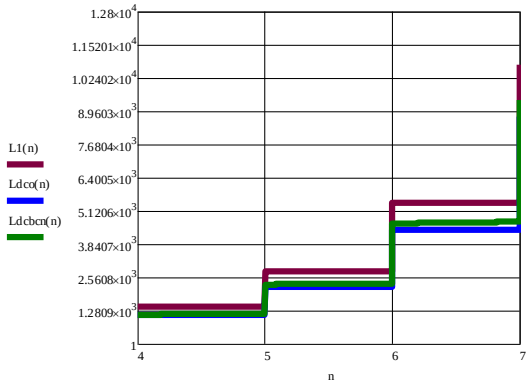


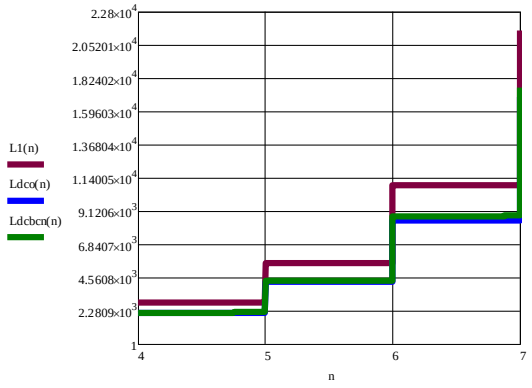
Figure 10. Single disjunctive block for n arguments

The block Fig. 10 performs the OR function of the CCNF elements.

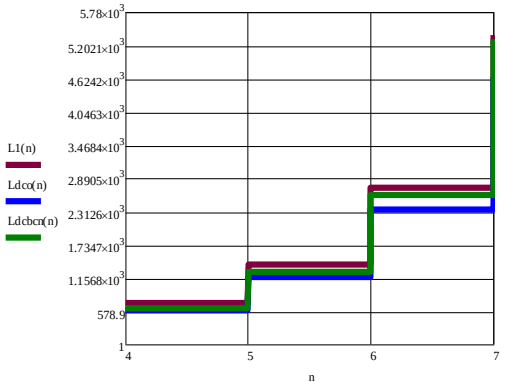
The comparison shows the advantages of the proposed device in the implementation of systems of functions that depend on the same variables. in Comparative curves of m function realization according to (19),(20),(21) in Mathcad shows Fig.11.



a) r=3;m=8



b) r=3;m=16



c) r=3;m=4

Figure 11. Comparison of the m LUT, DC-LUT-O and DC-LUT-BCN (Ldcbcn) with different n,r.

It easy to see, that Ldco is better, than Ldcbcn (and L1, of course). Let get relation L1/ Ldco:

$$\delta = \frac{m[(2^{n+1}-2)+8\cdot 2^n+4n+2\cdot\sum_{i=1}^{\left\lfloor\frac{n}{r}\right\rfloor}2^{n-ir}+2]}{2\cdot(2^{n+1}-2)+8\cdot 2^n+4n+2\cdot\sum_{i=1}^{\left\lfloor\frac{n}{r}\right\rfloor}2^{n-ir}+2+8m\cdot 2^n}. \quad (19)$$

The resulting expression (19) is a new scientific result, the use of which makes it possible to evaluate the advantages of a new technical solution. Curves of the expression (19) represents Fig.12.

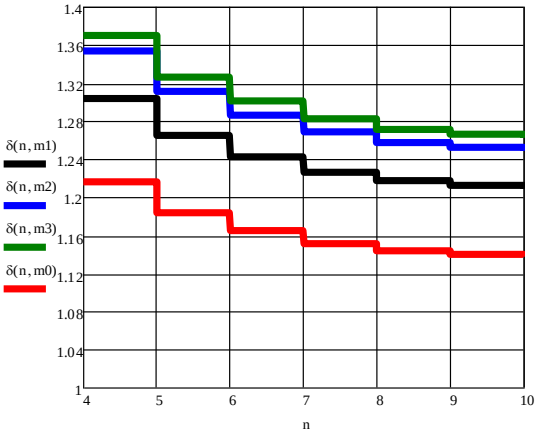


Figure 12. Curves of the relation L1/ Ldco; r=3; m0=4; m1=8; m2=16; m3=24;

Therefore, to greater m we get greater advantages of the DC LUT. If only single Decoder is produced, we get maximum profit – Fig.13.

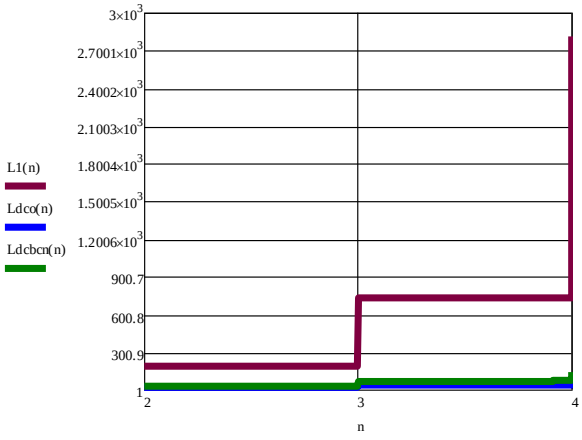


Figure 13. Only Single Decoder advantages

Combined DC-LUT O (Fig.8) and DC-LUT BCN (Fig.9) architecture we can get DC-LUT/BCN-O expression (20), (Fig. 14,15).

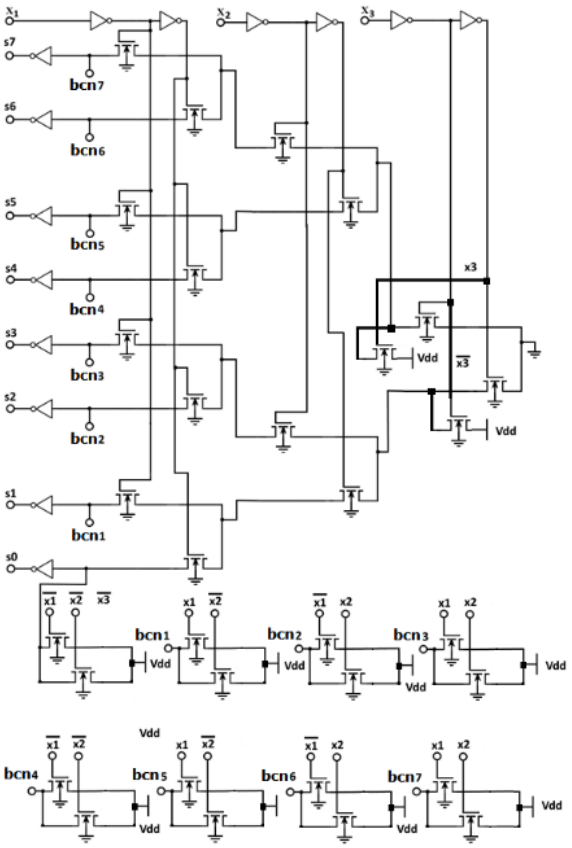
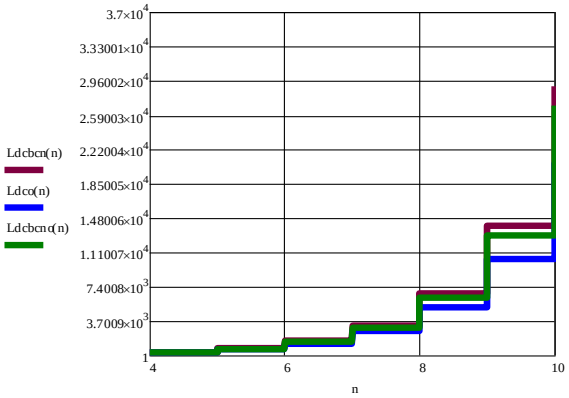


Figure 14. Combined 3-DC-LUT BCN-O with orthogonal by outputs (s0,s1,..,s6,s7) via only two variables

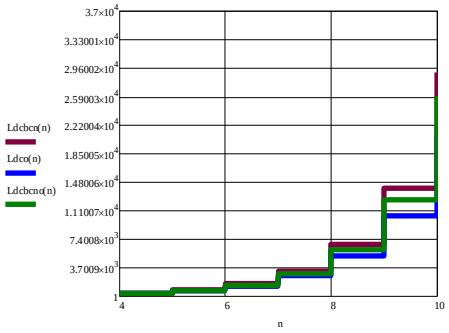
$$L_{n(\max r)-DC-LUT-BCN-O} = (2^{n+1}-2)+8\cdot 2^n + (n-j)\cdot 2^n + 4n + 2\cdot\sum_{i=1}^{\left\lfloor\frac{n}{r}\right\rfloor}2^{n-ir} + 2^{j+1}, \quad (20)$$

where, *j*-is the number of "O" variables.

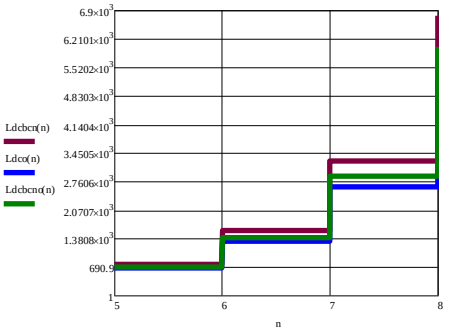
The comparison in Mathcad shows the advantages of the proposed device in the implementation of systems of functions that depend on the same variables.



a) j=2, n=4…10



b) j=3, n=4…10



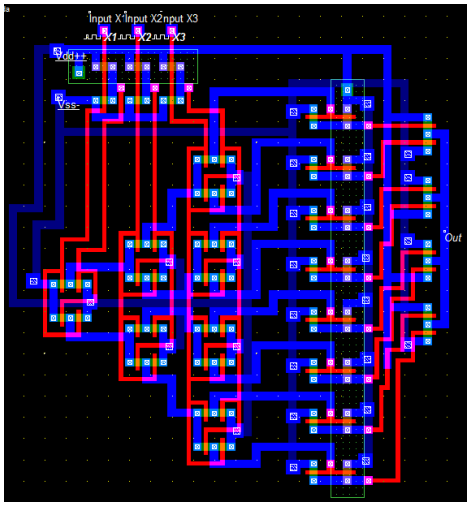
c) j=3, n=5…8

Figure 15. Comparison of the proposed DC-LUT-O(Ldco), DC-LUT-BCN (Ldcbcn), DC-LUT-BCN-O (Ldcbcno) at different j a) j=2, n=4…10; b) j=3, n=4…10; c) j=3, n=5…8

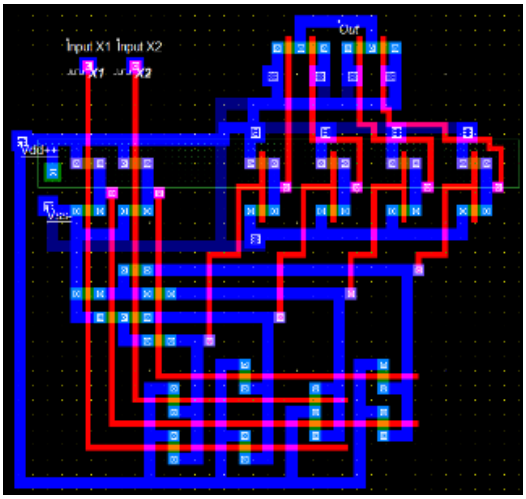
Therefore, DC-LUT-BCN loses to DC-LUT-O at the large n (Fig.15). However, estimates in the number of transistors are not enough, it is necessary to take into account the topology. Then we get layout simulation in Microwind CAD [17] using accessible transistors model [18].

IV. DC LUT LAYOUT SIMULATION

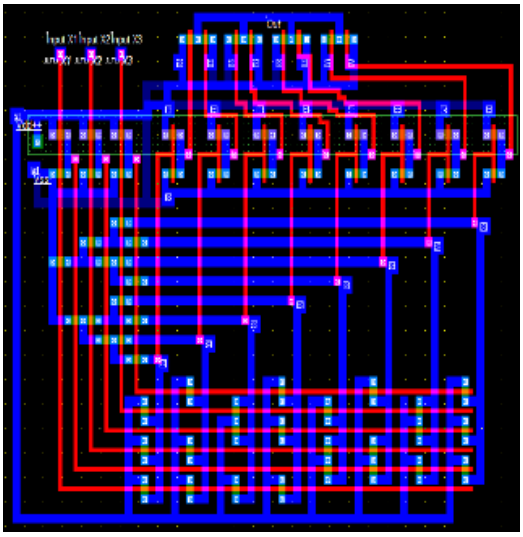
Proposed DC-LUT layout simulation in Microwind CAD [17] with Spice MOSFET Model BSIM4.8, 65nm [18] is shown in Fig. 16.



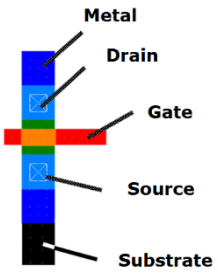
a)



b)



c)



d)

Figure 16. Proposed 3-DC-LUT layout: a) 3-DC-LUT-O; b) 2-DC-LUT-BCN; c) 3-DC-LUT-BKN; d) single transistor

Authors proposes Adaptive DC-LUT too. The device can, depending on the setting, perform the functions of both LUT and DC-LUT. Adaptive DC-LUT layout simulation in Microwind [17], [18] is shown in Fig. 17.

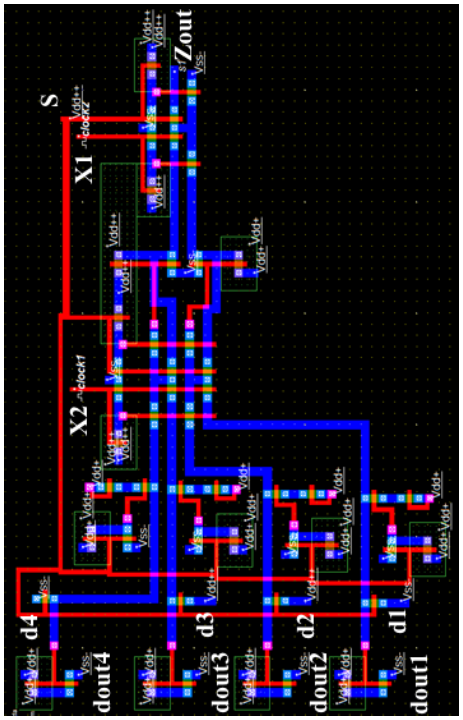


Figure 17. Proposed 2-ADC-LUT layout

Results of the simulation are shown in Table 1. 3-DC-LUT–BKN layout simulation for XOR is shown in Fig. 18. The layout simulation results confirm the efficiency of the proposed new technical solutions and allow you to choose the best options.

Common results of the proposed devices layout simulation in comparison with the known solution LUT are shown in Table 1.

We see the correct formation of a logical zero in cases (X1X2X3)=001, 010, 100,111 (Fig. 17).

Table 1. Results of the LUT/ DC-LUT layout simulation

№	Name	Layout Square S um^2	Power consumption Microwind (1V) In dynamic (uW)	Time delay T (ps)
1	1-LUT for single function	2,8	5.384	6
2	2-LUT for single function	3,8	7.297	13
3	3-LUT for single function	6	8.833	17
4	1-DC-LUT-O for system	6,2	10.327	8
5	2-DC-LUT-O for system	9	28.018	15
6	3-DC-LUT-O for system	19,1	62.552	20
7	1-DC-LUT-BKN for system	5	10.309	9
8	2-DC-LUT-BKN for system	11	27.377	16
9	3-DC-LUT-BKN for system	22,2	58.40	21

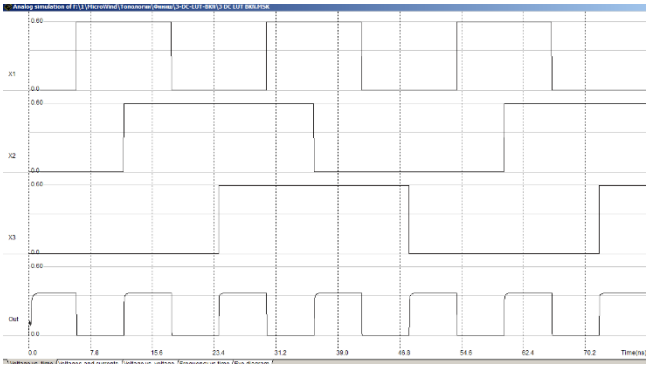


Figure 18. 3-DC-LUT–BKN simulation, waveform of XOR (X1X2X3)

We see that DC LUT outperforms the well-known solution LUT when implementing a system of decoding functions (8): for example 3-DC-LUT-O has 19,1 um^2 against 3-LUT 6*8=48 um^2.

V. CONCLUSION

The scientific novelty of the research lies in the fact that authors propose new gate named DC LUT for the realization of the logic function systems in FPGAs. The existing LUT elements implement only one function, so there are as many of them as there are functions of a given number of arguments. The complexity estimates are obtained and investigated, confirming the effectiveness of the new element. Detailed comparative modeling was performed in the systems of circuit simulation Maltisim and MicroWind. Most effect is achieved for the simple n-decoder, when each from 2^n function includes only one product term. Layout simulation proves workability of the proposed devices. DC-LUT-BCN loses to DC-LUT-O in transistors quantity at the large n, but has more layout square and better in dynamic power consumption. In time delay these variants are almost equal. Combining DC-LUT-BNC and DC-LUT-O technology allows achieving better characteristics. Proposed adaptive gate – ADC-LUT gate can be considered as a further development of ALM and possible model of the reversible computing [19], [20] for the Fredkin Gate implementation in reversible computing. The proposed elements allow to create advanced FPGAs of a new generation for embedded systems and on-board computers too.

References

[1] S. Brown, *Architecture of FPGAs and CPLDs: A Tutorial*. [Online]. Available at: https://www.ece.iastate.edu/~zambreno/classes/cpre583/documents/BroRos96A.pdf (accessed: 04.06.2021).

[2] A. Kaviani, S. Brown, *Hybrid FPGA architecture*, Available at: https://https://www.eecg.utoronto.ca/~brown/papers/fpga96-kaviani.pdf (accessed 04.06.2021)

[3] C. H. Ho, C. W. Yu, P. H. W. Leong, W. Luk and S. J. E. Wilton, "Domain-specific hybrid FPGA: Architecture and floating point applications," *Proceedings of the 2007 International Conference on Field Programmable Logic and Applications*, 2007, pp. 196-201, https://doi.org/10.1109/FPL.2007.4380647.

[4] M. Ebrahimi, R. Sadeghi and Z. Navabi, "LUT Input reordering to reduce aging impact on FPGA LUTs," *IEEE Transactions on*

Computers, vol. 69, no. 10, pp. 1500-1506, 2020. https://doi.org/10.1109/TC.2020.2974955.
[5] N. Mehta, *An Ultra-Low-Energy, Variation-Tolerant FPGA Architecture using Component-Specific Mapping*, Ph.D. Thesis, California Institute of Technology, [Online]. Available at: http://thesis.library.caltech.edu/7226/1/Nikil-Mehta-2013.pdf.
[6] C. Chiasson, *Optimization and Modeling of FPGA Circuitry in Advanced Process Technology*, Master Thesis, University of Toronto, 2013. [Online]. Available at: https://tspace.library.utoronto.ca/bitstream/1807/42733/3/Chiasson_Charles_RE_201311_MASc_thesis.pdf.
[7] Field Programmable Gate Arrays, [Online]. Available at: http://www.eng.auburn.edu/~nelson/courses/elec4200/FPGA/FPGAoverview.pdf.
[8] FPGA Architecture White Paper – Altera. [Online]. Available at: https://www.altera.com/en_US/pdfs/literature/wp/wp-01003.pdf.
[9] 7 Series FPGAs Data Sheet: Overview, [Online]. Available at: https://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf.
[10] S. Tyurin, "A Quad CMOS gates checking method," *International Journal of Computing*, vol. 18, issue 3, pp. 258-264, 2019. https://doi.org/10.47839/ijc.18.3.1518.
[11] A. Drozd, M. Drozd, O. Martynyuk, M. Kuznietsov, "Improving of a circuit checkability and trustworthiness of data processing results in LUT-based FPGA components of safety-related systems," *CEUR Workshop Proceedings*, vol. 1844, pp. 654–661, 2017.
[12] A. Drozd, M. Drozd, M. Kuznietsov, "Use of natural LUT redundancy to improve trustworthiness of FPGA design," *CEUR Workshop Proceedings*, vol. 1614, pp. 322–331, 2016.
[13] S. Gao, D. Al-Khalili, N. Chabini, "An improved BCD adder using 6-LUT FPGAs," Proceedings of the IEEE 10th International New Circuits and Systems Conference, NEWCAS, 2012, pp. 13-16. https://doi.org/10.1109/NEWCAS.2012.6328944.
[14] H. Gao, Y. Yang, G. Dong, "Theoretical analysis of effect of LUT size on area and delay of FPGA," 2005. [Online]. Available at: https://www.researchgate.net/publication/294490633_Theoretical_analysis_of_effect_of_LUT_size_on_area_and_delay_of_FPGA.
[15] V. S. Vinitha, R. K. Sharma, "An efficient LUT Design on FPGA for memory-based multiplication," *Iranian Journal of Electrical and Electronic Engineering*, no. 4, pp. 462–476, 2019.
[16] C. A. Mead, L. Conway, Introduction to VLSI Systems. [Online]. Available at: https://www.betterworldbooks.com/product/detail/Introduction-to-VLSI-Systems-9780201043587.
[17] N. Paydavosi, BSIM4v4.8.0 MOSFET Model -User's Manual 2013 Available at: http://bsim.berkeley.edu/BSIM4/BSIM480.zip.
[18] CAD MicroWind. [Online]. Available at: https://www.microwind.net.
[19] E. Fredkin and T. Toffoli, "Conservative logic," *International Journal of Theoretical Physics*, vol. 21, issue 3/4, pp. 219-253, 1982. https://doi.org/10.1007/BF01857727.
[20] K. Morita, "Reversible logic gates," In: Theory of Reversible Computing. Monographs in Theoretical Computer Science. An EATCS Series. Springer, Tokyo: https://doi.org/10.1007/978-4-431-56606-9_4.
[21] T. Toffoli, "Reversible computing," In: de Bakker J., van Leeuwen J. (eds) Automata, Languages and Programming. ICALP 1980. Lecture Notes in Computer Science, 1980, vol 85. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-10003-2_104.

Drs, Prof. SERGEY F. TYURIN, *graduated from Perm High Command-Engineering Military School of Rocket Forces at 1975. PhD in Computer Science, Kharkov High Command-Engineering Military School of Rocket Forces (USSR) at 1991. DrS in Computer Science, Perm High Command-Engineering Military School of Rocket Forces (Russia), 1998. Now he works as Professor at the Department of Automation and Telemechanic of Perm National Research Polytechnic University and as Professor at the Department of Software Computing Systems of Perm State National Research University. Research interests: methods and means of assessment and ensuring for reliability.*

Dr. Ruslan V. Vikhorev, *graduated Perm National Research Polytechnic University, 2014. Post graduated student at the Department of Automation and Telemechanic Perm National Research Polytechnic University, 2018. PhD Computer science, Perm National Research Polytechnic University, 2019. Now he works as a Design Bureau chief of the scientific and technical center, JSC "Perm scientific-industrial instrument making company".*