Original article
DOI: 10.21683/1729-2646-2017-17-2-11-16

Characteristic features of LUT setting codes of Intel FPGAs

Sergey F. Tyurin, *Perm National Research Polytechnic University, Perm, Russia*
Andrey S. Prokhorov, *Perm National Research Polytechnic University, Perm, Russia*


Sergey F. Tyurin


Andrey S. Prokhorov

Abstract. *State-of-the-art digital circuit design widely uses field programmable gate arrays (FPGAs), in which the functions of logic cells and their connections are set up. That is defined in the configuration file that is loaded in the configuration memory cells (static random access memory) of FPGA from external memory. The logic itself is implemented in the so-called LUTs (Look Up Tables), multiplexors that implement memory cells, are based on transmitting transistors and represents a tree that is activated by a specific variable collection. The setting is multiplexor data, therefore logical (switching) function values for the specific collection are transmitted to the tree output. As it turns out, the associated LUT setting code can be decoded and used for analyzing synthesis results in Quartus II by Altera that has been acquired by Intel. Now Intel also specializes in FPGA production. The article considers an example of the synthesis of a simple combinational finite state machine that implements the so-called majority function (2 out of 3). This function equals 1 if the majority of variables equals 1. Majority function implementation diagram is synthesized in Quartus II that builds a special BDF (Block Diagram/Schematic File) file. The resulting diagram is examined with Map Viewer. In the appropriate diagram, LUT (Logic Cell Comb) setting codes for implementation of the specified function are set forth in the form of four-digit hexacodes. Decoding is shown for setting codes for logic cells of FPGA LUT type that describe the content of the respective truth tables of functions that depend on the input variable machine. The article shows the code changes in the process of diagram optimization by Quartus II with possible modification of the variables sequence order and correspondence with the inputs of a four-input LUT without modifications to the logical function. If Stratix IIGX FPGA is used that has the so-called adaptive logic modules (ALM) with 6 inputs, Quartus II uses 64-bit codes (eight-digit hexacodes). Respective coding is also examined in this paper.*

Keywords: *combinational machine, majority function (2 out of 3), logic cells, LUT (Look Up Table), FPGA (Field Programmable Gate Array), Logic Cell Comb, adaptive logic module (ALM).*

For citation: *Tyurin SF, Prokhorov AS. Characteristic features of LUT setting codes of Intel FPGAs. Dependability 2017; 2: 11-16. DOI: 10.21683/1729-2640-2017-17-2-11-16*

1. Introduction

Field Programmable Gate Arrays (FPGA) are based on random access memory (RAM) units called LUT (Look Up Table) [1-3]. The RAM units contain configuration information in the form of truth tables of the required logical functions.

The QuartusII system by Altera (US) that manufactures the FPGA allows synthesizing finite state machines. The machine is defined not only by a diagram in the form of BDF (Block Diagram/Schematic File), but also in the VHDL, Verilog, AHDL and other hardware description languages, as well as the State Machine File machine graph. QuartusII generates setting codes of logic cells.

Of interest is the decoding of this configuration data (Logic Cell Comb) and their comparison with specified logical functions of the machine. Let us define a simple machine and examine the associated setting codes.

2. Resulting truth tables for a four-input LUT

Let us assume that it is required to synthesize the implementation diagram for the three-variable switching function (SF) no. 132. Let us build a truth table (Figure 1).

Variables			BC	f(abc)	
a	b	c			
0	0	0	0	0	2^0
0	0	1	1	0	2^1
0	1	0	2	0	2^2
0	1	1	3	1	2^3
1	0	0	4	0	2^4
1	0	1	5	1	2^5
1	1	0	6	1	2^6
1	1	1	7	1	2^7

Fig. 1. SF truth table no. 132_{10}

This SF no. 132_{10} is a so-called majority function that in disjunctive normal form (DNF) is as follows:

$$f(abc) = ab \vee bc \vee ac = \overline{\overline{ab} \vee \overline{bc} \vee \overline{ac}} = \overline{\overline{ab} \cdot \overline{bc} \cdot \overline{ac}}.$$

Let us manually build the diagram of the form BDF (Block Diagram / Schematic File), Figure 2.

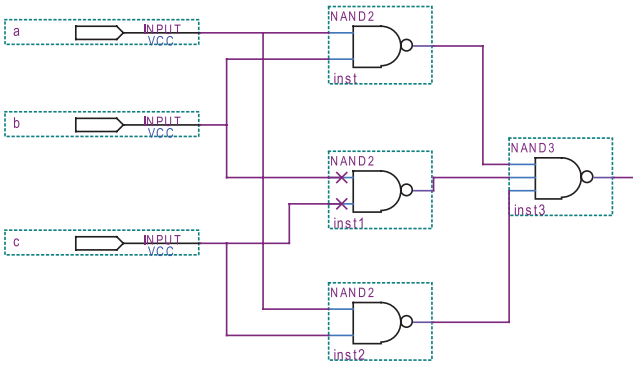

Fig. 2. Basic circuit diagram of the majority function in the form of QuartusII BDF (Block Diagram / Schematic File)

In Figure 2, the diagram inputs are marked as bus "+" of the V_{cc} power supply unit, the inputs are "pulled up" to the voltage, i.e. at the inputs in the initial state there are logical units. By FPGA-compiling the project (e.g. EP2-C5AF256A7) we obtain a report file (Figure 3).

Flow Status	In progress - Sat Jan 23 12:05:46 2016
Quartus II Version	9.0 Build 132 02/25/2009 SJ Web Edition
Revision Name	Lab1
Top-level Entity Name	Lab1
Family	Cyclone II
Device	EP2C5AF256A7
Timing Models	Final
Met timing requirements	N/A
Total logic elements	1 / 4,608 (< 1 %)
Total combinational functions	1 / 4,608 (< 1 %)
Dedicated logic registers	0 / 4,608 (0 %)
Total registers	0
Total pins	4 / 158 (3 %)
Total virtual pins	0
Total memory bits	0 / 119,808 (0 %)
Embedded Multiplier 9-bit elements	0 / 26 (0 %)
Total PLLs	0 / 2 (0 %)

Fig. 3. Diagram compilation results in the EP2C5A-F256A7 FPGA

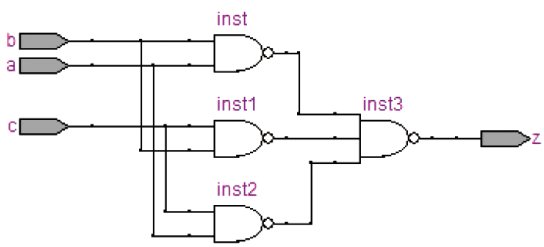

Fig. 4. RTL file

Let us analyze the report produced by Map Viewer. The RTL (register transfer level) diagram is shown in Figure 4.

The RTL diagram is practically identical to BDF and does not contain configuration information, but it is present in the report of the Technology Map Viewer in the hexacode form (Post Mapping, after the allocation of FPGA cells in the "map") (Figure 5).

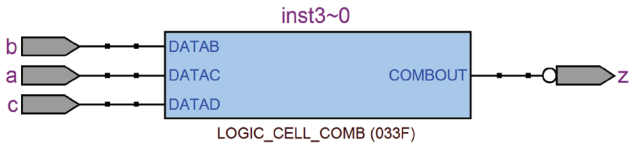

Fig. 5. Technology Map Viewer (Post Mapping)

Thus, QuartusII by Altera has "packed" our diagram into a single FPGA logic cell. Let us decode the 033F setting code of the logic cell LOGIC_CELL_COMB (Figure 6) by comparing the input variables with the 4-input logic cell inputs.

D	C	B	A	033F code	f(abc)
c	a	b	~		
0	0	0	0	1	0
0	0	0	1	1	0
0	0	1	0	1	0
0	0	1	1	1	0
0	1	0	0	1	0
0	1	0	1	1	0
0	1	1	0	0	1
0	1	1	1	0	1
1	0	0	0	1	0
1	0	0	1	1	0
1	0	1	0	0	1
1	0	1	1	0	1
1	1	0	0	0	1
1	1	0	1	0	1
1	1	1	0	0	1
1	1	1	1	0	1

Fig. 6. 033F decoding – Post Mapping

As we can see, the coding, as it should, begins with the higher orders of the truth table, where the inputs of the logic cell are in the order D, C, B, A, while the variables are in the order c, a, b. As Figure 5 shows a z output inversion, the 033F code is an inversion of the desired majority function f(abc). As we can see (Figure 5), input A is not used because the function depends on tree variables and QuartusII chose to use only the inputs B, C and D.

Let us now examine Technology Map Viewer (Post Fitting, i.e. after connections optimization), Figure 7.

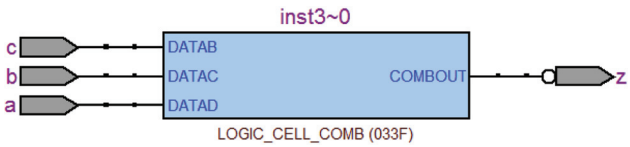

Fig. 7. Technology Map Viewer (Post Fitting)

Figure 7 shows that input connections have changed, which probably is the optimization of connections. Let us decode the code with the new variable connections.

D	C	B	A	033F code	f(abc)
a	b	c	~		
0	0	0	0	1	0
0	0	0	1	1	0
0	0	1	0	1	0
0	0	1	1	1	0
0	1	0	0	1	0
0	1	0	1	1	0
0	1	1	0	0	1
0	1	1	1	0	1
1	0	0	0	1	0
1	0	0	1	1	0
1	0	1	0	0	1
1	0	1	1	0	1
1	1	0	0	0	1
1	1	0	1	0	1
1	1	1	0	0	1
1	1	1	1	0	1

Fig. 8. 033F decoding – Post Fitting

The 033F coding does not change, because the majority function does not depend on the variables sequence order. Thus (from top to bottom) 033F is acquired.

3. Acquisition of the truth tables for a six-input ALM

Now, let us define a more complex Stratix IIGX FPGA that has the so-called adaptive logic modules (ALM) with not 4, but 6 inputs [4-6]. We obtain a report (Figure 9).

We can see that the report in Figure 9 features adaptive LUTs (ALUTs), their coding is 64-bit, i.e. 16 hexadecimal digits (Figure 10).

As previously, there is an output inversion. Let us decode part by part the setting, code E8E8E8E8E8E8E8E8, that also should be an inversion of the majority function. We will take into consideration the additional inputs F and E (Figure 11).

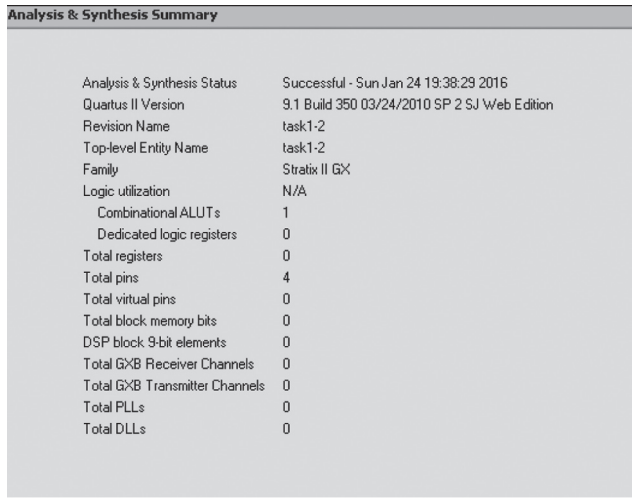

Fig. 9. Stratix IIGX FPGA compilation results

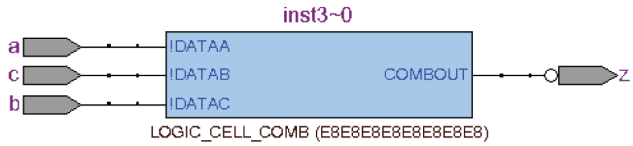

Fig. 10. Technology Map Viewer (Post Mapping) for a Stratix IIGX FPGA diagram

F	E	D	C	B	A	E8E8	f(abc)
~	~	~	b	c	a		
0	0	0	0	0	0	0	1
0	0	0	0	0	1	0	1
0	0	0	0	1	0	0	1
0	0	0	0	1	1	1	0
0	0	0	1	0	0	0	1
0	0	0	1	0	1	1	0
0	0	0	1	1	0	1	0
0	0	0	1	1	1	1	0
0	0	1	0	0	0	0	1
0	0	1	0	0	1	0	1
0	0	1	0	1	0	0	1
0	0	1	0	1	1	1	0
0	0	1	1	0	0	0	1
0	0	1	1	0	1	1	0
0	0	1	1	1	0	1	0
0	0	1	1	1	1	1	0

Fig. 11. Decoding of a part of E8E8 code (Post Mapping), first part of the truth table

However, in this case the majority function is not implemented. An inversion is implemented instead. Why? Let us try the coding from low orders (Figure 12).

Thus, the noncompliance of the ALUT settings with the required function can be explained by the "inverted" coding, i.e. from low orders (from the top of the table in Figure 12). That might be caused by the large size of the truth table.

F	E	D	C	B	A	E8E8	f(abc)
~	~	~	b	c	a		
0	0	0	0	0	0	1	0
0	0	0	0	0	1	1	0
0	0	0	0	1	0	1	0
0	0	0	0	1	1	0	1
0	0	0	1	0	0	1	0
0	0	0	1	0	1	0	1
0	0	0	1	1	0	0	1
0	0	0	1	1	1	0	1
0	0	1	0	0	0	1	0
0	0	1	0	0	1	1	0
0	0	1	0	1	0	1	0
0	0	1	0	1	1	0	1
0	0	1	1	0	0	1	0
0	0	1	1	0	1	0	1
0	0	1	1	1	0	0	1
0	0	1	1	1	1	0	1

Fig. 12. Decoding of a part of E8E8 code from low orders (Post Mapping), first part of the ALUT truth table

The remaining three parts of the truth table are shown in Figures 13 – 15.

F	E	D	C	B	A	E8E8	f(abc)
~	~	~	b	c	a		
0	1	0	0	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	0	1	0	1	0
0	1	0	0	1	1	0	1
0	1	0	1	0	0	1	0
0	1	0	1	0	1	0	1
0	1	0	1	1	0	0	1
0	1	0	1	1	1	0	1
0	1	1	0	0	0	1	0
0	1	1	0	0	1	1	0
0	1	1	0	1	0	1	0
0	1	1	0	1	1	0	1
0	1	1	1	0	0	1	0
0	1	1	1	0	1	0	1
0	1	1	1	1	0	0	1
0	1	1	1	1	1	0	1

Fig. 13. Decoding of a part of E8E8 code from low orders (Post Mapping), second part of the ALUT truth table

Let us examine the coding of Post Fitting for a Stratix IIGX FPGA diagram (Figure 16).

We can see that now the variables have "shifted" towards the higher orders F and E. Therefore, the code is different. Let us verify the implementation of the specified function (Figure 17).

F	E	D	C	B	A	NOT	f(abc)
~	~	~	b	c	a	f(abc)	
1	0	0	0	0	0	1	0
1	0	0	0	0	1	1	0
1	0	0	0	1	0	1	0
1	0	0	0	1	1	0	1
1	0	0	1	0	0	1	0
1	0	0	1	0	1	0	1
1	0	0	1	1	0	0	1
1	0	0	1	1	1	0	1
1	0	1	0	0	0	1	0
1	0	1	0	0	1	1	0
1	0	1	0	1	0	1	0
1	0	1	0	1	1	0	1
1	0	1	1	0	0	1	0
1	0	1	1	0	1	0	1
1	0	1	1	1	0	0	1
1	0	1	1	1	1	0	1

Fig. 14. Decoding of a part of E8E8 code from low orders (Post Mapping), third part of the ALUT truth table

F	E	D	C	B	A	NOT	f(abc)
~	~	~	b	c	a	f(abc)	
1	1	0	0	0	0	1	0
1	1	0	0	0	1	1	0
1	1	0	0	1	0	1	0
1	1	0	0	1	1	0	1
1	1	0	1	0	0	1	0
1	1	0	1	0	1	0	1
1	1	0	1	1	0	0	1
1	1	0	1	1	1	0	1
1	1	1	0	0	0	1	0
1	1	1	0	0	1	1	0
1	1	1	0	1	0	1	0
1	1	1	0	1	1	0	1
1	1	1	1	0	0	1	0
1	1	1	1	0	1	0	1
1	1	1	1	1	0	0	1
1	1	1	1	1	1	0	1

Fig. 15. Decoding of a part of E8E8 code from low orders (Post Mapping), forth part of the ALUT truth table

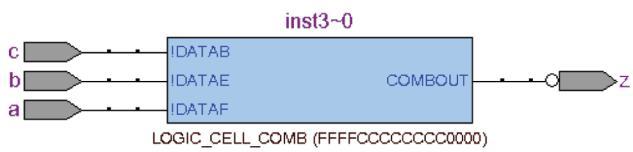

Fig. 16. Technology Map Viewer (Post Fitting) for a Stratix IIGX FPGA diagram

F a	E b	D ~	C ~	B c	A ~	FFFF code	f(abc)
0	0	0	0	0	0	1	0
0	0	0	0	0	1	1	0
0	0	0	0	1	0	1	0
0	0	0	0	1	1	1	0
0	0	0	1	0	0	1	0
0	0	0	1	0	1	1	0
0	0	0	1	1	0	1	0
0	0	0	1	1	1	1	0
0	0	1	0	0	0	1	0
0	0	1	0	0	1	1	0
0	0	1	0	1	0	1	0
0	0	1	0	1	1	1	0
0	0	1	1	0	0	1	0
0	0	1	1	0	1	1	0
0	0	1	1	1	0	1	0
0	0	1	1	1	1	1	0

Fig. 17. Decoding of the first part the ALUT FFFF code from low orders (Post Fitting)

F a	E b	D ~	C ~	B c	A ~	CCCC code	f(abc)
0	1	0	0	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	0	1	0	0	1
0	1	0	0	1	1	0	1
0	1	0	1	0	0	1	0
0	1	0	1	0	1	1	0
0	1	0	1	1	0	0	1
0	1	0	1	1	1	0	1
0	1	1	0	0	0	1	0
0	1	1	0	0	1	1	0
0	1	1	0	1	0	0	1
0	1	1	0	1	1	0	1
0	1	1	1	0	0	1	0
0	1	1	1	0	1	1	0
0	1	1	1	1	0	0	1
0	1	1	1	1	1	0	1

Fig. 18. Decoding of the second part the ALUT CCCC code from low orders (Post Fitting)

F a	E b	D ~	C ~	B c	A ~	CCCC code	f(abc)
1	0	0	0	0	0	1	0
1	0	0	0	0	1	1	0
1	0	0	0	1	0	0	1
1	0	0	0	1	1	0	1
1	0	0	1	0	0	1	0
1	0	0	1	0	1	1	0
1	0	0	1	1	0	0	1
1	0	0	1	1	1	0	1
1	0	1	0	0	0	1	0
1	0	1	0	0	1	1	0
1	0	1	0	1	0	0	1
1	0	1	0	1	1	0	1
1	0	1	1	0	0	1	0
1	0	1	1	0	1	1	0
1	0	1	1	1	0	0	1
1	0	1	1	1	1	0	1

Fig. 19. Decoding of the third part the ALUT CCCC code from low orders (Post Fitting)

F a	E b	D ~	C ~	B c	A ~	0000 code	f(abc)
1	1	0	0	0	0	0	1
1	1	0	0	0	1	0	1
1	1	0	0	1	0	0	1
1	1	0	0	1	1	0	1
1	1	0	1	0	0	0	1
1	1	0	1	0	1	0	1
1	1	0	1	1	0	0	1
1	1	0	1	1	1	0	1
1	1	1	0	0	0	0	1
1	1	1	0	0	1	0	1
1	1	1	0	1	0	0	1
1	1	1	0	1	1	0	1
1	1	1	1	0	0	0	1
1	1	1	1	0	1	0	1
1	1	1	1	1	0	0	1
1	1	1	1	1	1	0	1

Fig. 20. Decoding of the forth part the ALUT 0000 code from low orders (Post Fitting)

4. Conclusions

Given the above, the hexacodes, LUT configuration data (LOGIC_CELL_COMB), can be transformed into truth tables of the respective logical functions. The variables sequence order (the "base" of variables) can be arbitrary and changes when the diagram is optimized (when changing from Post Mapping to Post Fitting), yet the function itself remains unchanged.

The coding discrepancy (LOGIC_CELL_COMB) between the Stratix IIGX FPGAs with a 6-variable adaptive logic module (64 bits, 16 hexadecimal digits) and FPGAs with a 4-variable LUT (16 bits, 4 hexadecimal digits) can be explained by the use of "reverse" coding. In this case, LOGIC_CELL_COMB starts with not the higher, but the lower orders of the truth table. The above decoding should complement laboratory classes of the implementation of digital machines in the Quartus system [7].

References

1. Ugriumov EP. Tsyfrovaya skhemotekhnika: ouchebnoie posobie [Digital circuit design: a study guide]. Saint Petersburg: BHV-Petersburg; 2004 [in Russian].

2. Tsybin S. Programmiruemaia kommutatsia PLIS: vzgliad iznutri [Software switching of FPGA: a look from the inside], <http://www.kit-e.ru/articles/plis/2010_11_56.php> [accessed on 16.12.2014] [in Russian]/

3. An Ultra-Low-Energy, Variation-Tolerant FPGA Architecture Using Component-Specific Mapping [Electronic resource], <http://thesis.library.caltech.edu/7226/> [accessed on 11.11.14].

4. Zolotukha R, Komolov D. Stratix III — novoye semeystvo FPGA firmy Altera [Statix III, a new FPGA family by Altera], <http://kit-e.ru/assets/files/pdf/2006_12_30.pdf> [accessed on 28.11.2015] [in Russian].

5. Ispolzovanye resursov PLIS Stratix III firmy Altera pri proektirivanii mikroprotsessornykh yader [Use of the resources of Stratix III FPGA by Altera in the design of microprocessor cores], <http://www.kit-e.ru/articles/plis/2010_2_39.php> [accessed on 27.11.2015] [in Russian].

6. Logic Array Blocks and Adaptive Logic Modules in Stratix III Devices, <https://www.altera.com.cn/content/dam/altera-www/global/zh_CN/pdfs/literature/hb/stx3/stx3_siii51002.pdf> [accessed on 29.11.2015] [in Russian].

7. Tyurin SF, Gromov OA, Grekov AV. Realizatsia tsifrovykh avtomatov v systeme Quartus firmy Atera: laboratorny praktikum [Implementation of digital machines in the Quartus system by Altera: a laboratory practicum]. Perm: PNRPU Publishing; 2011 [in Russian].

About the authors

Sergey F. Tyurin, Honourable Inventor of the Russian Federation, Doctor of Engineering, Professor of Automation and Remote Control, Perm National Research Polytechnic University. Perm, Russia, e-mail: tyurinsergfeo@yandex.ru

Andrey S. Prokhorov, post-graduate, Department of Automation and Remote Control, Perm National Research Polytechnic University. Perm, Russia, e-mail: npoxop007@yandex.ru.

Received on 06.03.2016