A new FPGA with 4/5-input LUT and optimized carry chain*

Mao Zhidong(毛志东), Chen Liguang(陈利光), Wang Yuan(王元), and Lai Jinmei(来金梅)†

State Key Laboratory of ASIC and System, Fudan University, Shanghai 201203, China

Abstract: A new LUT and carry structure embedded in the configurable logic block of an FPGA is proposed. The LUT is designed to support both 4-input and 5-input structures, which can be configured by users according to their needs without increasing interconnect resources. We also develop a new carry chain structure with an optimized critical path. Finally a newly designed configurable scan-chain is inserted. The circuit is fabricated in 0.13 μm 1P8M 1.2/2.5/3.3 V logic CMOS process. The test results show a correct function of 4/5-input LUT and scan-chain, and a speedup in carry performance of nearly 3 times over current architecture in the same technology at the cost of an increase in total area of about 72.5%. Our results also show that the logic utilization of this work is better than that of a Virtex II/Virtex 4/Virtex 5/Virtex 6/Virtex 7 FPGA when implemented using only 4-LUT and better than that of a Virtex II/Virtex 4 FPGA when implemented using only 5-LUT.

Key words: FPGA; configurable logic block; 4/5-input LUT; carry chain optimization; scan-chain
DOI: 10.1088/1674-4926/33/7/075009 **EEACC:** 1265A

1. Introduction

The configurable logic block is the key cell of an FPGA, basically including LUTs, carry chains and flip-flops, used to implement most combinatorial and sequential functions. Mainstream commercial FPGA devices are all based upon LUT structure and a simple carry scheme, similar to that found in Altera Stratix and Cyclone series FPGAs[1, 2], lattice EC/ECP series FPGAs[3] and Xilinx Virtex II/Virtex 4/Virtex 5/Virtex 6/Virtex 7 series FPGAs[4−8].

According to some academic research[9, 10], an LUT with 4 to 6 inputs demonstrates the best area and delay performance. However, the traditional 4-input LUT structure leads to a low logic density and a lack of configuration flexibility, reducing the utilization of interconnect resource when configured as relatively complex logic functions. The Xilinx's newly designed Virtex 7 series FPGAs and Virtex 6 FPGAs have employed the 6-input LUT structure: a configurable logic block contains two slices and each slice includes four 6-input LUTs with two outputs for each LUT[4, 5]. Virtex 5 FPGAs have the same LUT structure as Virtex 7/Virtex 6 FPGAs but have fewer storage elements inside each slice (eight in Virtex7/Virtex 6 and four in Virtex 5)[4−6]. The Xilinx Virtex II and Virtex 4 series FPGAs use the traditional 4-input LUT structure[7, 8]. In Ref. [11] a 3-input LUT based logic cell is addressed, which increases logic density when compared with a traditional 4-input LUT. The work in Refs. [12, 13] provides a heterogeneous architecture for an FPGA logic element, which is an extended K-LUT and proves to be area-efficient. This paper presents a new LUT, which can be configured between 4 and 5 input structures, increasing configuration flexibility and logic density without increasing the complexity of the interconnect structure.

The FPGA is becoming the driving force behind a new computing paradigm, in which the FPGA realization of fast and compact binary adders relies on hardware carry chains[14]. The work in Ref. [15] presents a redesigned standard ripple carry chain to reduce the number of logic levels in each cell and achieves a speedup in carry performance of 3.8 times over current architectures. The work in Ref. [16] proposes a LUT-based carry chain with high flexibility. In Ref. [17] a carry chain optimized for implementing multipliers is addressed, which is compatible with both 3-input as well as 4-input LUTs.

According to Refs. [18, 19], simple carry chain based adders continue to be the fastest implementations as they exploit dedicated routing and fast logic gates to reduce the critical path of arithmetic operations. In this paper we propose a new carry structure with fast and slow paths. Our design is focused on the carry-select mechanism and the optimization of the carry propagation path that goes through the carry chain.

2. 4/5-input configurable LUT

2.1. New LUT structure

The size of the LUT in a traditional SLICE of a configurable logic block is fixed, such as in the 4-input LUT shown in Fig. 1[1, 2]. The SLICE has two 4-input LUTs, which can be configured as a 5-input LUT (F5 as the output) at most by adding BX as the fifth input. This structure is the trade-off among area, speed and power with relatively fewer inputs, saving a large number of interconnect resources. But its low logic density and configuration inflexibility reduces the effective utilization of interconnect resources when configured as complex logic functions.

Figure 2 shows the new 4/5-input LUT structure. Two new 4-input LUTs, F-new and G-new, are introduced and the output multiplexers are expanded by adding ports F4AUX connected to the outputs of the new LUTs. The 5-input LUT function is realized by sharing existing input ports F[4:1] (or G[4:1]) and reusing BX (or BY) as the fifth input for the LUT.

* Project supported by the National High Technology Research and Development Thematic Program of China (No. 2012AA012001).
† Corresponding author. Email: jmlai@fudan.edu.cn
Received 29 December 2011, revised manuscript received 29 February 2012

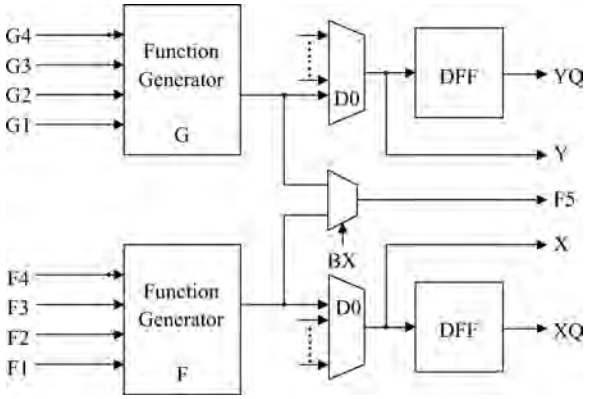

Fig. 1. Fixed 4-input LUT.

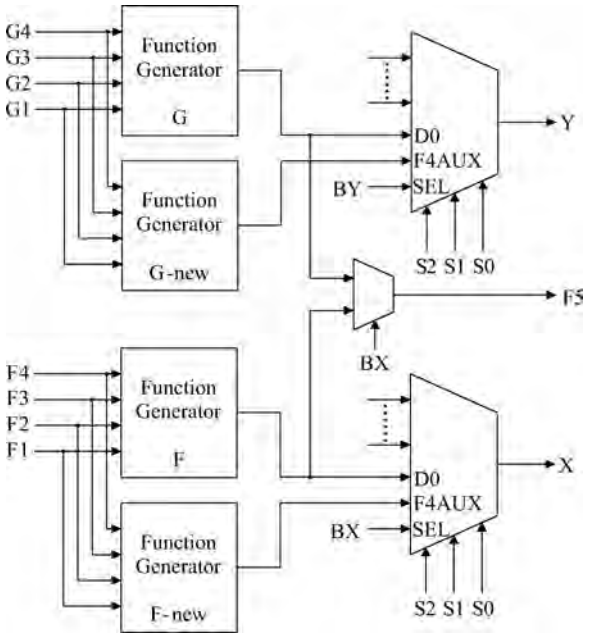

Fig. 2. 4/5-input configurable LUT.

This new structure provides an extra support for a 5-input LUT instead of being fixed during configuration. It can be configured between 4-input and 5-input LUTs by different logic combinations of configuration bits S2, S1 and S0. Moreover, the 4/5-input configurable LUT does not change the number of ports of a SLICE, thus saving interconnect resources while increasing the logic density.

2.2. Expanded output MUX

Figure 3 is a detailed circuit diagram of the expanded output multiplexer. $X(Y)1$ is the decoded result of signals other than LUT outputs (F(G) or F(G)-new) which appear on $X(Y)0$ through MUX1 controlled by configuration bit S1 and reused input BX (or BY).

MUX2 is controlled by configuration bit S2 and its reverse signal S2B decides the type of $X(Y)$ output, which is described in Table 1.

$X(Y)$ is used as the LUT output when S2 is logic 0. When S2 is logic 1, $X(Y)$ acts as other function outputs, which are determined in the Decoder module. Configuration bit S1 chooses between 4-input and 5-input LUT structures when $X(Y)$ is used as the LUT output. As shown in Table 2, when S1 is logic 0, 4-input LUT is in use and the value of BX (or BY) does not matter. When S1 is logic 1, BX (or BY) chooses between two LUT results (F and F-new (or G and G-new)), activating the 5-input LUT.

Table 1. MUX2 truth table.

S2	S2B	$X(Y)$
0	1	$X(Y)0$
1	0	$X(Y)1$

Table 2. MUX1 truth table.

S1	BX(BY)	$X(Y)0$
0	X	D0
1	0	D0
1	1	F4AUX

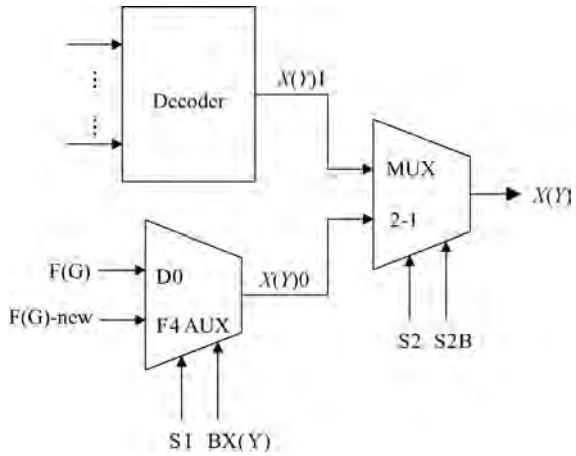

Fig. 3. Expanded output MUX.

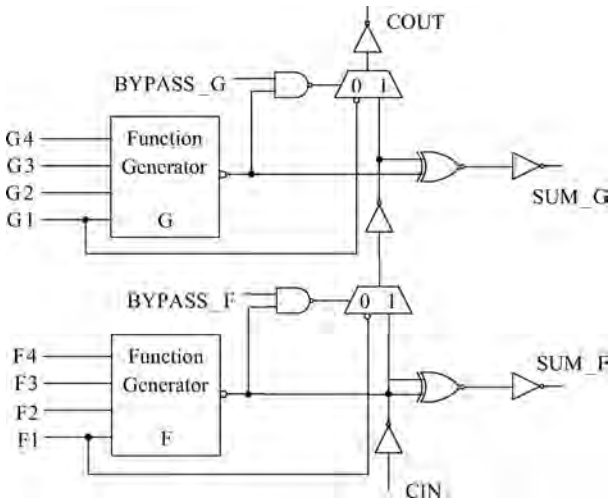

Fig. 4. Traditional carry structure.

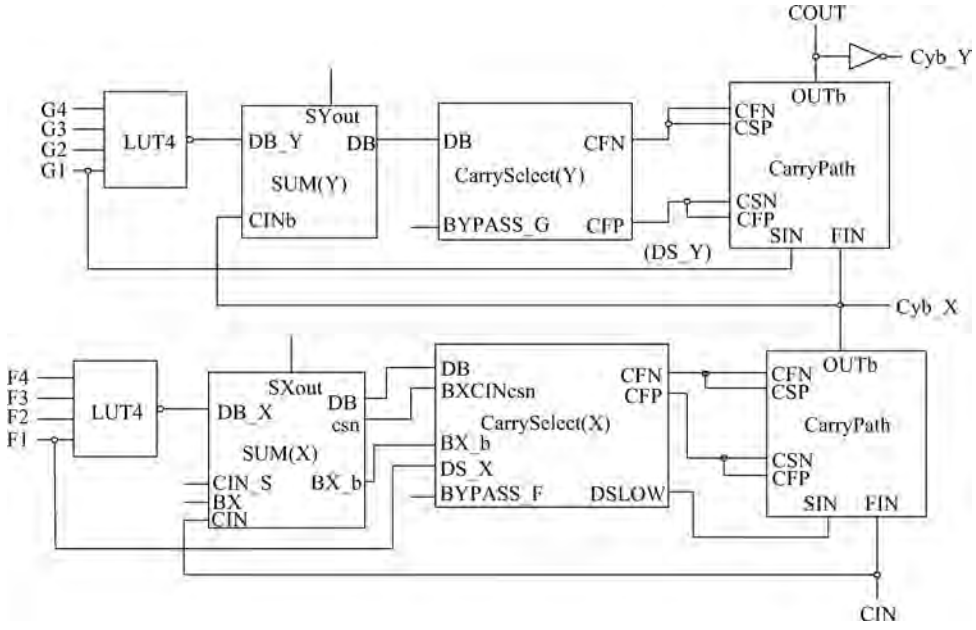

Fig. 5. Carry chain with fast/slow paths.

3. Carry chain with fast/slow paths

3.1. Traditional carry chain

Arithmetic operations are often found when designing FPGAs. In order to save area and optimize delay, specific carry chain circuits are embedded in current commercial FPGAs according to the demands on computation. A traditional carry structure is shown in Fig. 4[1, 2].

For a typical 1-bit full-adder, A and B are two operands, CIN is the carry input, SUM is the result of the arithmetic operation and COUT is the carry output. The logic expressions are as follows:

$$\mathrm{SUM} = \mathrm{A} \oplus \mathrm{B} \oplus \mathrm{CIN}, \tag{1}$$

$$\mathrm{COUT} = (\overline{\mathrm{A} \oplus \mathrm{B}}) \ \& \ \mathrm{A} + (\mathrm{A} \oplus \mathrm{B}) \ \& \ \mathrm{CIN}. \tag{2}$$

A, B correspond to F1, F2 or G1, G2 in Fig. 5. BYPASS_F(G) is a configuration bit that controls the MUX selection signal that lies in the carry chain. When it is logic 1, the result of LUT appears on the output of the NAND gate, realizing regular arithmetic operation and carry propagation. When it is logic 0, the output of the NAND gate is fixed to logic 1, forcing the MUX to directly output the carry input.

The traditional carry structure shows two paths for carry signal: F1 (G1) or the bypass of the carry input. The characteristic of a carry chain demands that the LUT used to implement arithmetic operations is on the same column and the carry input is right next to the preceding carry output, which makes the placement and routing complicated because not all LUTs in an FPGA are configured to perform computation. For example, there could be another LUT placed between the two LUTs, as shown in Fig. 4, configured as any function irrelevant to arithmetic operation, in which a bypass operation is needed. So the carry propagation path becomes the critical path that affects the performance of computation.

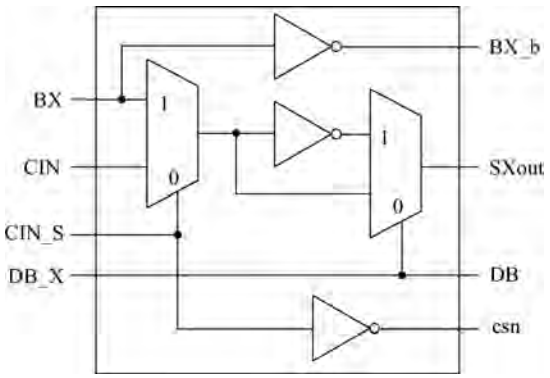

Fig. 6. Circuit of SUM(X).

3.2. New carry chain structure

Figure 5 shows a newly introduced carry chain structure, which is divided into a slow path and a fast path (SIN port and FIN port in the CarryPath module). The SUMX(Y) module finishes the sum operation and passes the sum of F1 and F2 (or G1 and G2) to the CarrySelect$X(Y)$ module, which decides whether the carry output comes from the fast path or the slow path according to the value of the sum received. The following will present more detailed circuit design methods of each module in the carry chain, taking part X of the signal path as an example. Figure 6 shows the circuit of the SUM(X) module, which realizes the following functions:

$$\mathrm{DB} = \overline{\mathrm{F1} \oplus \mathrm{F2}}, \tag{3}$$

$$\mathrm{SXout} = \overline{\mathrm{DB}} \ \& \ \mathrm{CIN} + \mathrm{DB} \ \& \ \overline{\mathrm{CIN}} = \overline{\mathrm{D} \oplus \mathrm{CIN}}. \tag{4}$$

3.3. Fast/slow path selection mechanism

Table 3 describes the rule that determines the value of the carry output (COUT). When $\mathrm{A} \oplus \mathrm{B} = 0$, COUT equals one of

Table 3. Determination of COUT.

A	B	D = A ⊕ B	CIN		COUT
0	0	0	*X*	0	One of operands
1	1	0	*X*	1	One of operands
0	1	1	0	0	CIN
1	0	1	1	1	CIN

Table 4. Fast/slow path selection mechanism.

Path	Signal	BX or CIN	Active Condition
Fast	CIN	BXCINcsn = 1	BYPASS_F = 1 & DB = 0
			BYPASS = 0
Slow	BX	BXCINcsn = 0	BYPASS_F = 1 & DB = 0
			BYPASS = 0
	DS_X	–	BYPASS_F = 1 & DB = 1

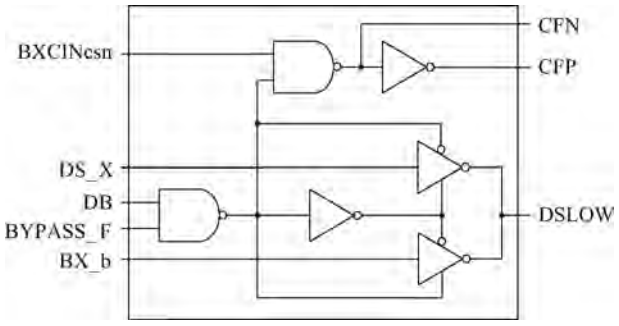

Fig. 7. Circuit of CarrySelect(X).

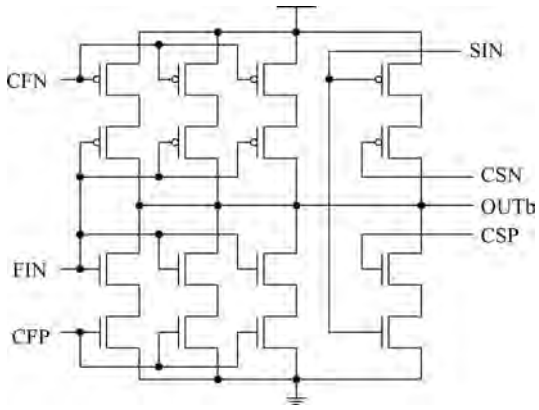

Fig. 8. Circuit of CarryPath.

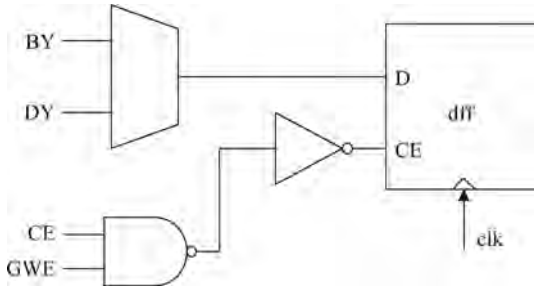

Fig. 9. Original Datapath of DFF.

the arithmetic operands (A or B) regardless of what value CIN is. When A ⊕ B = 1, COUT equals the carry input (CIN).

According to Table 3, the principle of selecting between fast and slow paths is described in Table 4. The fast path only propagates the preceding carry output (i.e. the current carry input) while the slow path passes the carry input of the LSB (least significant bit) and the carry signal that comes from the current operand (DS_X, i.e. F1). The CarrySelect(*X*) circuit module designed according to Table 3 is shown in Fig. 7, in which the logic expression of the fast/slow selection signal CFP is as follows:

$$\mathrm{CFP} = \mathrm{BXCINcsn} \ \& \ \overline{\mathrm{DB} \ \& \ \mathrm{BYPASS\_F}}. \qquad (5)$$

The internal circuit of the CarryPath module is presented in Fig. 8. All the transistors used in this circuit are low voltage threshold MOS transistors so that the carry propagation speed is optimized. When CFP is logic 1 (CSN = 1, CFN = CSP = 0), the fast path is activated and the slow path is disabled, FIN (i.e. CIN) is propagated to the output. When CFP is logic 0 (CSN = 0, CFN = CSP = 1), the fast path is closed and the slow path is open enabling SIN (i.e. BX or DS_X) to pass through. As to part *Y* of the signal path, the analysis is the same.

4. Scan-chain insertion

4.1. Modification of the flip-flop's datapath

A flip-flop's datapath of a slice contains the combinational output DY (D1) of the LUT and an input signal BY (D0) from the outside, as shown in Fig. 9. Taking advantage of the rich resource of flip-flops in an FPGA, we modified the datapath structure of every flip-flop by adding the scan-chain function to implement serial-scan testing.

Figure 10 shows the expansion we made to dmux based on the original datapath structure. Two sets of signals (Scan_Data and Scan_Enable) are added and configuration bits are expanded to three srams (S2, S1, S0) to support different configuration modes. The scan-chain works as follows. When Scan_Enable = 0, it works as a regular flip-flop with DY or BY connected to the data input of DFF and CE controlled by the user. When Scan_Enable = 1, the scan function is enabled. The control output of dmux is forced to logic 0, thus setting CE of DFF to logic 1 to make sure scan data can be loaded into flip-flops during scan testing. The logic 0 of dmux's control output also sets the SR input of DFF to logic 0 to guarantee the correct operation of scan testing by preventing users from accidentally resetting the system.

Note that the Scan_Data signal includes three kinds of inputs: BY (SD1), F4/G4 (SD2) and the output of the previous DFF (SD3) so that the scan-chain can be built conveniently without loss of flexibility. Table 5 shows the truth table of dmux in different configuration modes.

4.2. Overview of the scan-chain structure

Figure 11 shows an overview of the scan-chain structure. There are two scan-chains in a configurable logic block. One is upward and the other is downward. The two scan-chains share the Scan_Enable signal. In an FPGA, specific circuits in IOB modules around the configurable logic blocks can be programmed to determine whether a longer scan-chain needs to be built.

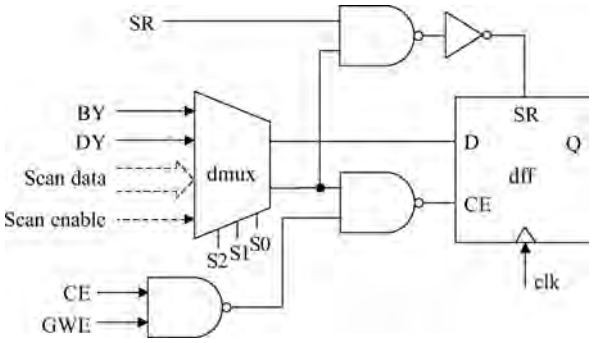

Fig. 10. New Datapath of DFF.

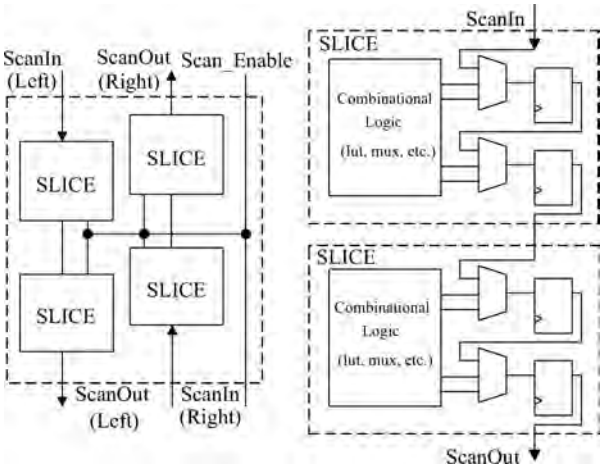

Fig. 11. Overview of the scan-chain.

Table 5. Truth table of dmux.

S2	S1	S0	ScanE	Q
0	1	1	*X*	D0
1	1	1	*X*	D1
0	*X*	*X*	0	D0
1	*X*	*X*	0	D1
X	0	0	1	SD1
X	0	1	1	SD2
X	1	1	1	SD3

Figure 12 shows the simulation result of the left scan-chain in a configurable logic block. As we can see, during scan test mode, the 4 flip-flops clock in the scan data and when the scan test is disabled they clock in regular input data.

5. Experimental results

5.1. Layout implementation

Figure 13 shows the layout of this newly designed configurable logic block with two new LUTs for each slice and an improved carry chain structure. For a clear view of the function modules inside a configurable logic block, the metal lines above the second layer of aluminum lines are not shown in the figures.

The SLICE inside a black frame represents the circuit of a slice without the D-flip-flop unit. A configurable logic block has four slices (and an additional two rows of srams for config-

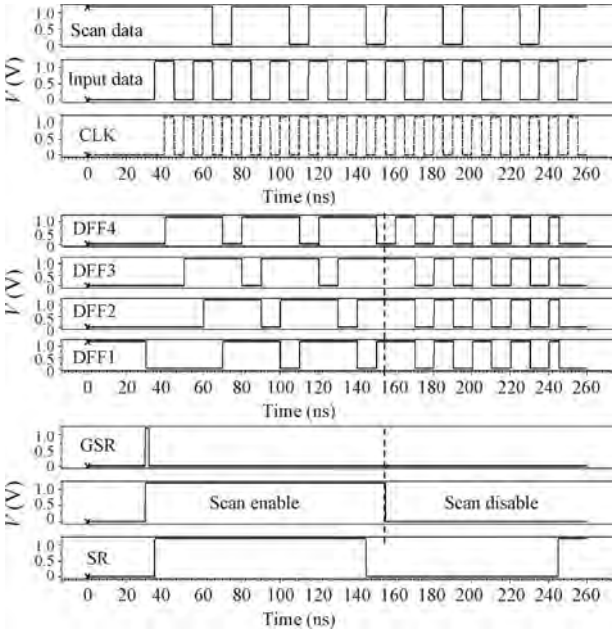

Fig. 12. Scan-chain simulation result.

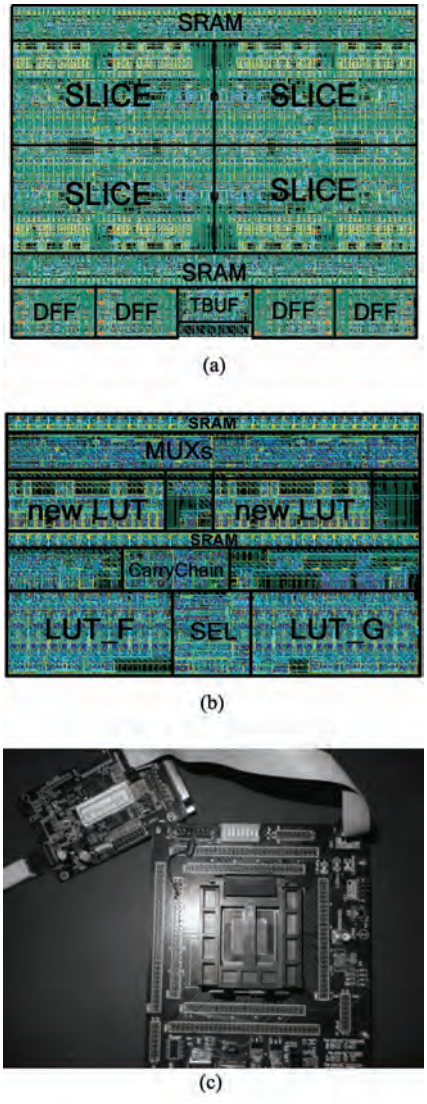

Fig. 13. (a) Layout of slice. (b) Layout of configurable logic block. (c) FPGA test board.

Table 6. Functional test of the configurable logic block.

Function	Description	Result
lut1	Logic functions of 1 to 5 input look-up table, xor-gate and flip-flop	Pass
lut2		
lut3		
lut4		
lut5		
ram1	Distributed ram	Pass
ram2		
shift	Shift register	Pass
bxq	bx input and xq output	Pass
cinyq	Datapath of carry chain	Pass
f8mux1	Multiplexers inside the slice	Pass
f8mux2		
f × 3		
f × 4		
xand	and gate	Pass
Latch	Configuration of latch	Pass
sop	Sum of product	Pass
cy0fg	cy0f and cy0g multiplexers	Pass
sr	Different configuration modes of set/reset, sync/async and enable control of flip-flop	Pass
srb		
ce		
ceb		
Adder	4 bit adder	Pass
Counter	4 bit counter	Pass

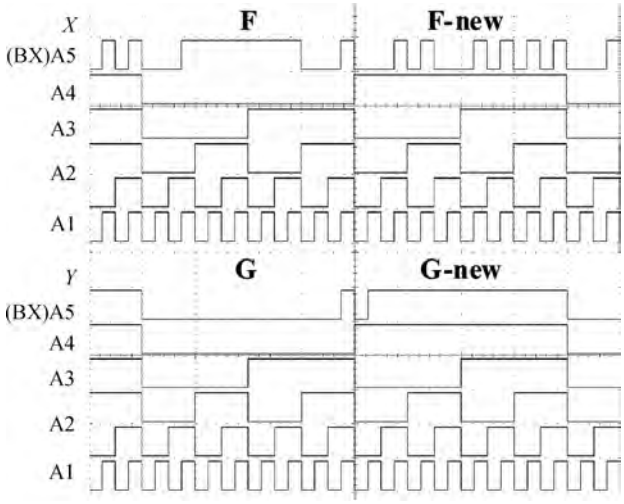

Fig. 14. 5-input LUT test result.

uration), four D-flip-flops and a TBUF. Inside the slice a new pair of LUTs is inserted between the multiplexers and the carry chain module. The area of the two new LUTs is less than one fourth of the area of the original slice. The layout of the configurable logic block appears as a clear and neat rectangle and shows a placement of modularity.

5.2. Functional test result

Figure 14 shows the test results of the 5-input LUT configured as:

$$X = \{\overline{A5} \ \& \ [(A4 \oplus A3)|(A2 \ \& \ A1)]\}| \{A5 \ \& \ [A4|(A3 \oplus A2)] \ \& \ A1\}, \quad (6)$$

$$Y = [\overline{A5} \ \& \ (A4 \ \& \ A3 \ \& \ A2 \ \& \ A1)]| [A5 \ \& \ (A4|A3|A2|A1)]. \quad (7)$$

The results demonstrate that the 5-input LUT structure functions correctly.

Table 6 shows examples of the passed functional tests of the configurable logic block presented in this paper. The test results show that the newly designed configurable logic block corrects functions correctly.

5.3. Performance test result of carry chain

In a previous work we designed an FPGA using the traditional 4-input LUT structure and the critical path of the carry chain was not optimized. The work presented in this paper employs a new 4/5-input configurable LUT structure, thus adding an additional 5-LUT feature. In addition, we attempt to optimize the critical path of the carry chain.

Table 7 shows a comparison of test results among three FPGA chips (Virtex-4 implemented in a 90 nm copper CMOS process, the previous work and this work implemented in the same 0.13 μm 1P8M 1.2/2.5/3.3 V logic CMOS process). The test results of this work show a new feature of 5-LUT which the previous work does not have, and a speedup in carry performance of nearly 3 times that achieved in the previous work at the cost of an increase in total area of about 72.5%. Table 7 also shows that the carry delay through a slice of the Xilinx Virtex-4 XCSX35 FPGA is reduced by about 30% compared with that of this work, which is proportional to the decrease in λ from a 0.13 μm process to a 90 nm process.

Table 7. Performance test results.

Parameter	Virtex-4 XC4VSX35	Previous work	This work
1. Technology	90 nm	0.13 μm	0.13 μm
2. 5-LUT feature	No	No	Yes
3. Carry delay through a slice (ns)	0.070 ↑ 0.071 ↓	0.302 ↑ 0.312 ↓	0.109 ↑ 0.101 ↓
4. Total area of a configurable logic block (μm^2)	—	180 × 80 μm^2	184 × 138 μm^2

Table 8. Number of LUTs in a configurable logic block.

Parameter	4-LUT	5-LUT	6-LUT
This work	16	8	2
Previous work	8	4	2
Virtex II/4	8	4	2
Virtex 5/6/7	8	8	8

Table 9. Number of configurable logic blocks needed to implement a specific function using only 4-LUTs, 5-LUTs or 6-LUTs.

Parameter	4-LUT	5-LUT	6-LUT
This work	n	2n	8n
Previous work	2n	4n	8n
Virtex II/4	2n	4n	8n
Virtex 5/6/7	2n	2n	2n

Note: n represents the number of configurable logic blocks needed in this work.

Table 10. Comparison of logic utilization.

Parameter	4-LUT	5-LUT	6-LUT
This work versus Virtex II/4	Better	Better	Same
This work versus Virtex 5/6/7	Better	Same	Worse

5.4. Logic utilization comparison

Table 8 shows the number of equivalent 4-LUTs, 5-LUTs or 6-LUTs inside a configurable logic block of this work, previous work mentioned above, Xilinx Virtex II/Virtex 4 FPGAs and Virtex 5/Virtex 6/Virtex 7 FPGAs. We can see that the number of 4-LUTs in this work is twice as much as that of the Virtex II/Virtex 4/Virtex 5/Virtex 6/Virtex 7 FPGAs, the number of equivalent 5-LUTs in this work is the same as that of the Virtex 5/Virtex 6/Virtex 7 FPGAs and twice that of the Virtex-II/Virtex 4 FPGAs, the number of equivalent 6-LUTs in this work is the same as that of Virtex II/Virtex 4 FPGAs and one-fourth of that of Virtex 5/ Virtex 6/Virtex 7 FPGAs.

Table 9 shows the number of configurable logic blocks needed to implement a specific function using only 4-LUTs, 5-LUTs or 6-LUTs among these different series of FPGAs. When implemented using only 4-LUTs, the number of configurable logic blocks needed in this work is reduced by half compared with that of Virtex II/Virtex 4/Virtex 5/Virtex 6/Virtex 7 FPGAs. When implemented using only 5-LUTs, the number is the same as that for Virtex 5/Virtex 6/Virtex 7 FPGAs and half of that for Virtex II/Virtex 4 FPGAs. When implemented using only 6-LUTs, the number is the same with that of Virtex II/Virtex 4 FPGAs and four times as much as that of Virtex 5/Virtex 6/Virtex 7 FPGAs.

Table 10 shows a comparison of logic utilization between this work and Xilinx Virtex II/Virtex 4/ Virtex 5/Virtex 6/Virtex 7 series FPGAs. We can see that the logic utilization of this work is better than that of Virtex II/Virtex 4/Virtex 5/Virtex 6/Virtex 7 FPGAs when implemented using only 4-LUT and better than that of Virtex II/Virtex 4 FPGAs when implemented using only 5-LUT.

6. Conclusion

A 4/5-input configurable LUT is proposed which increases logic utilization and configuration flexibility. The interface of the configurable logic block remains the same so that the structural complexity of the interconnect is not increased. We also developed a new carry chain with fast and slow paths, the experimental results of which show a significant speedup in carry performance of nearly 3 times over current commercial architecture. Finally, we inserted a newly designed scan-chain to increase the testability of the chip. The newly added 5-LUT feature and the optimization in carry performance contribute to an increase in total area of about 72.5%.

References

[1] Altera Corp. Cyclone/Cyclone-II Device Handbook, 2005
[2] Altera Corp. Stratix/Stratix-II/Device Handbook, 2005
[3] Lattice Semiconductor Corp. LatticeECP/EC Family Data Sheet, May 2005
[4] Xilinx, Inc. 7 Series FPGAs Configurable Logic Block User Guide, 2011
[5] Xilinx, Inc. Virtex-6 FPGA Configurable Logic Block User Guide, 2009
[6] Xilinx, Inc. Virtex-5 FPGA User Guide, 2009
[7] Xilinx, Inc. Virtex-4 FPGA User Guide, 2008
[8] Xilinx, Inc. Virtex-II Platform FPGAs: Complete Data Sheet, 2005
[9] Ahmed E, Rose J. The effect of LUT and cluster size on deep-submicron FPGA performance and density. IEEE Trans Very Large Scale Integration Syst, 2004, 12(3): 288
[10] Farooq U, Marrakchi Z, Mrabet H, et al. The effect of LUT and cluster size on a tree based FPGA architecture. International Conference on Reconfigurable Computing and FPGAs, 2008: 115
[11] Chen Liguang, Wang Yabin, Wu Fang, et al. Design and implementation of an FDP chip. Journal of Semiconductors, 2008, 29(4): 713
[12] Anderson J H, Wang Q. Improving logic density through synthesis-inspired architecture. International Conference on Field Programmable Logic and Applications (FPL), 2009: 105
[13] Anderson J H, Wang Q. Area-efficient FPGA logic elements: architecture and synthesis. 16th Asia and South Pacific Design Automation Conference (ASP-DAC), 2011: 369
[14] Nguyen H D, Pasca B, Preusser T B. FPGA-specific arithmetic optimizations of short-latency adders. International Conference on Field Programmable Logic and Applications (FPL), 2011: 232

[15] Hauck S, Hosler M M, Fry T W. High-performance carry chains for FPGA's. IEEE Trans Very Large Scale Integration Syst, 2000, 8(2): 138
[16] Lodi A, Chiesa C, Campi F, et al. A flexible LUT-based carry chain for FPGAs. Proceedings of the International Symposium on Circuits and Systems, 2003, 5: 133
[17] Jindal V, Agarwal A. Carry circuitry for LUT-based FPGA. Proceedings of 17th International Conference on VLSI Design, 2004: 731
[18] Preusser T B, Spallek R G. Enhancing FPGA device capabilities by the automatic logic mapping to additive carry chains. International Conference on Field Programmable Logic and Applications (FPL), 2010: 318
[19] Zicari P, Perri S. A fast carry chain adder for Virtex-5 FPGAs. 15th IEEE Mediterranean Electrotechnical Conference, 2010: 304