RC-005

A^2LUT: Abridged Adaptive LUT Architecture

Masahiro Iida† **Ken Taura**† **Masahiro Koga**† **Kazuki Inoue**†
Motoki Amagasaki† **Toshinori Sueyoshi**†

1. Introduction

Field Programmable Gate Arrays (FPGAs) are an important building block of reconfigurable systems and also have a cost-efficiency as an alternative to Application Specific Integrated Circuits (ASICs). However, FPGAs based on Look-up Tables (LUTs) doesn't have enough area efficiency as compared with standard-cell-based ASICs, because that a FPGA has numerous circuit resources, such as configuration memory bits, interconnection switches and wires. Especially the LUTs are built to store the truth tables of logic functions, and it can emulate the equivalent of a logic gate, therefore. K inputs LUTs has 2^K configuration memory bits which can express 2 powers of 2^K logic patterns. This is the critical factor of an area for FPGAs, because the influence of a big amount of configuration memory bits on the area and power is very large.

Many fine-grain logic cells have been proposed as alternative FPGA devices. To reduce the number of configuration memory bits, Y. Hu et al. [1] designed a heterogeneous programmable logic block using a combination of LUTs and macrogates which comprised several combinational logic gates. They also proposed a synthesis flow for such programmable logic blocks. However, placement and routing evaluations are not performed. We have studied the architecture of variable-grain logic cell (VGLC), which have the features of both coarse-grained and fine-grained logic cells [2]. VGLC delivers a good performance until the technology mapping phase. Conversely, a routing problem arises owing to input/output pin overhead in VGLC. Furthermore, we proposed a small-memory logic cell, COGRE [3], in order to reduce the FPGA area. Their approach produces circuits with very good area efficiency. However, the delay performance is not enough as compared with other fine grain logic blocks, because of the logic step numbers increase.

Based on these previous studies, we focus on the following two points to overcome these issues.

Functionality: The total number of logic blocks can be reduced by increasing the functionality per unit logic block. However, it is important to inhibit increasing both the logic area and the number of configuration memory bits in a logic block.

Adaptability: It leads to improvement of the implementation efficiency by improving with the adaptability of the logical block. For example, the implementation efficiency is improved dramatically if devices can support LUT of plural granularity, such as MCMG-LUT [4] and ALM (Adaptive Logic Module) [5], in one logic block.

In this paper, we propose new logic block architecture in order to reduce both FPGA chip area and delay.

The paper is organized as following. P-equivalence class [6] is described in Section 2. In Section 3, we discuss the concept of our logic block architecture, and describe the logic cell design. Section 4 and Section 5 introduces performance evaluation and result. In Section 6, we overview our developed prototype chip. Finally, Section 7 concludes this paper.

2. P-equivalence class

We explain a permutation equivalence (P-equivalence) class [6] as the preparations before proposing a new logical block. P-equivalence class is defined as follows.

Definition 1: The minterm expansion of an n-variable function is $f(x_1, x_2, ..., x_n) = c_0 \cdot \overline{x}_1\overline{x}_2 \cdot \cdot \cdot \overline{x}_n \vee c_1 \cdot \overline{x}_1\overline{x}_2 \cdots x_n \vee \cdots \vee c_{2^n-1} \cdot x_1x_2 \cdots x_n$, where $c_0, c_1, ..., c_{2^n-1} \in \{0, 1\}$. The binary digit c_j is called the coefficient of the $j-th$ minterm, $j-thcoefficient$. The 2^n bit binary number $c_0c_1...c_{2^n-1}$ is the binary number representation of f. To denote a binary number, a subscripted 2 is used after it.

Example 1: Consider the three-variable function $f(x_1, x_2, x_3) = \overline{x}_1\overline{x}_2\overline{x}_3 \vee x_1$. The binary number representation of f is 10001111_2

Definition 2: Two functions f and g are P-equivalent if g can be obtained from f by permutation of the variables [7][8]. $f \overset{P}{\sim} g$ denotes that f and g are P-equivalent. P-equivalent functions from a *P-equivalence class* of functions.

Example 2: Consider the three functions: $f_1(x_1, x_2, x_3) = \overline{x}_2\overline{x}_3 \vee x_1x_2x_3$,$f_2(x_1, x_2, x_3) = \overline{x}_1\overline{x}_3 \vee x_1x_2x_3$, and $f_3(x_1, x_2, x_3) = \overline{x}_1\overline{x}_2 \vee x_1x_2x_3$. Since $f_2(\mathbf{x_2}, \mathbf{x_1}, x_3) = \overline{x}_2\overline{x}_3 \vee x_1x_2x_3 = f_1(x_1, x_2, x_3)$, we have $f_1 \overset{P}{\sim} f_2$, and since $f_3(x_1, \mathbf{x_3}, \mathbf{x_2}) = \overline{x}_1\overline{x}_3 \vee x_1x_2x_3 = f_2(x_1, x_2, x_3)$,

†Graduate School of Science and Technology, Kumamoto University

Table 1: Number of P-representations in LUT.

# inputs of LUT	# of expressible functions	# of P-representation
3-LUT	256	80
4-LUT	65,536	3,984
5-LUT	about 4 billion	about 37 million

we have $f_2 \overset{P}{\sim} f_3$. Therefore, the functions f_1, f_2 and f_3 belong to the same P-equivalence class.

Definition 3: The function that has the smallest binary number representation among the functions of a P-equivalence class is the P-representative of the class.

Firstly, we classify the logic functions of LUT on the basis of the P-equivalence class. We show the number of logic functions that can implement to LUT in the number of each input and the number of P-representation in Table 1. In the case of the logic of three inputs, this result means what can implement all 256 kinds of functions if we can implement 80 kinds of P-representation or either the functions of P-equivalence class. Therefore, we can implement all functions of the K inputs by the number of configuration memory bits less than the necessary memory bits in order to express 2^{2^K} functions.

Secondly, we investigate the appearance ratio of the P-equivalence class in 6-LUT mapped net lists. We analyze the appearance ratio of logic functions on 117 MCNC benchmark circuits using P-equivalence class for classifying the logic functions [9]. We also perform technology mapping and investigate a kind of the implemented LUT functions. In order to avoid tool independence, gate-level net lists are trans-formed 6-LUT based net lists by two major mapping tools, FlowMap [10] and EMap [11]. Our experiment results show that the top 50 logic patterns account for 60% of the used P-representative in benchmarks, and the top 500 patterns cover 90% of the used P-representative. Similar P-representatives often tend to be used in FlowMap and EMap. The results show that only small portions of the P-equivalence class can cover large portions of the logic functions used to implement circuits. Assuming this analysis, we develop new logic block architecture and CAD tools for it.

3. Logic Cell Design

3.1 Principle of Abridging Configuration Bits

In reference [9], many of the employed logic functions were based on AND and OR gate, and the function binnary number consists of consecutive bits of 0 or 1. We reduce the circuit resource of LUT by abridging the same bit string to the configuration memory of 1 bit.

Table 2: Example of consecutive bits in 3-LUT.

Name	Boolean Expressions	Function binary
Func 1	$A \vee \overline{B} \cdot C$	0100_1111_2
Func 2	$\overline{A} \cdot \overline{B} \cdot \overline{C}$	1000_0000_2
Func 3	$\overline{A} \cdot B \cdot \overline{C}$	0010_0000_2

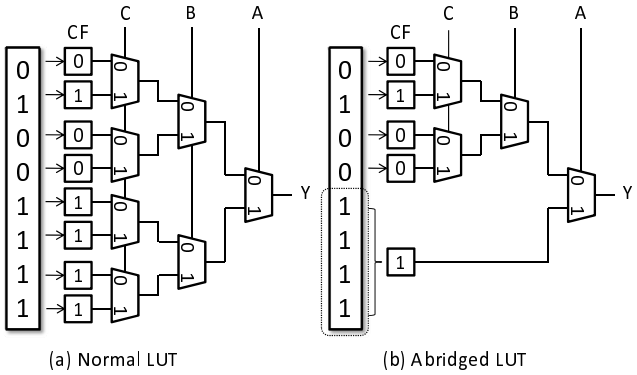


Figure 1: Principle of Abridged LUT.

To explain the principles of abridging of configuration bits by the cases of three 3-input functions shown in Table 2. Normal 3-LUT is shown in Figure 1 (a), uses 8-bit configuration memory in order to implement a logic function.

On the other hand, because a value of lower 4 bits is the same as for the logic functions in Table 2, we can assign its same value in Fig. 1(b) to one configuration memory. In that case, this performs the completely same behaviour as Fig. 1(a). As a result, it is possible to reduce the amount of circuit resource, such as configuration memory bits.

Of course, implementable circuits are decreased in number by reducing the configuration memory bits. We use the P-equivalence class, which we explained in the previous section, in order to hold back a fall of the logic cover rate. We decide an abridgment point of the configuration memory bits in the following procedures to be concrete.

1. Sort the logic functions in benchmark circuits in order that the appearance probability is high.
2. In each logic function, you can select a logic function including a lot of bit string to be common from P-equivalence class.
3. And search for an abridgment point of configuration memory bits that is covering a lot of higher logic functions of the incidence.

3.2 Abridged Adaptive LUT

We propose an abridged adaptive LUT (A^2LUT) by considering upper rank logic with P-equivalence

class. Figure 2 shows proposed LUT architecture. The A^2LUT has two 3-input LUT, extra two configuration memory bits M[8] and M[9], other three MUXes and several logic gates, six input pins and one output pin. The MUX, which is an output side, is used when 4 input logic is implemented the A^2LUT. Other two MUXes which are connected the combination logic output is used when 4 inputs, 5 input or 6 inputs LUT functions are switched. As well as 4-LUT, the A^2LUT can implement all of 4 input logic functions. Conventional 4-LUT has 16 configuration memory bits, while our A^2LUT has 18 configuration memory bits.

On the other hand, 5-LUT and 6-LUT requires 32 and 64 configuration memory bits, respectively. As 5 and 6 inputs function, the A^2LUT cannot cover all of P-representative, but it can be implemented the subset of P-representative which include high frequently used. In fact, the A^2LUT can cover 53.25% of 5 inputs logic functions and 51.79% of 6 inputs logic functions on 117 MCNC benchmark sets.

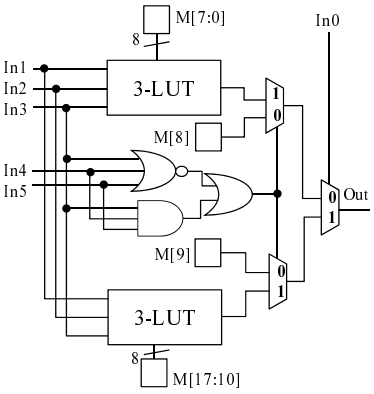
We show the mapping example to 6 inputs A^2LUT in Table 3. This list is top ten circuits of the P-representative of the high appearance rate in the result that mapped 117 kinds of MCNC benchmark circuits onto 6-LUT by FlowMap. As for the A^2LUT, only 18 bits have configuration memory, but we want to implement a circuit of 6-LUT(64 bits) by folding of configuration memory shown in Table 3(1). However, when we implement these circuits in the A^2LUT, we cannot implement by using the function binary in shown of P-representative. Accordingly, we can implement all circuits in this ranking by converting it into an adequate P-equivalence class. Table 3(2) shows the example that we can implement by the permutation. When we assign the logic variable (A,B,C,D,E,F) into the input port (in0, in1, in2, ..., in5) of the logic cell in this example, the function binary number of the P-representative (a) is not able to write into configuration memory of A^2LUT. Even so, if we swap A for D, and we update a truth table with it. It becomes (b). As a result, we become implementable for this circuit. From the above, we can implement more than 50% of the logic of 6-LUT that we investigated by the permutation of variable.

3.3 Cluster-Based Architecture

For performance improvement, conventional FPGAs introduce a cluster-based FPGA architecture [12]. In this evaluation, we decide the cluster architecture with size of 4, which are adopted in several conventional FPGAs. Figure 3 shows a logic cluster with 4 BLEs(short for a basic logic element), four feedbacks, and a fully connected local routing networks. Each BLE contains four 6-input A^2LUTs, D-FFs, and output-selection MUX. The number of cluster inputs (I) are determined from the following formula [13].

In0	In1	In2	In3	In4	In5	Out
0	*	*	0	0	0	M[3:0]
0	*	*	0	0	1	M[8]
0	*	*	0	1	0	
0	*	*	0	1	1	
0	*	*	1	0	0	
0	*	*	1	0	1	
0	*	*	1	1	0	
0	*	*	1	1	1	M[7:4]
1	*	*	0	0	0	M[13:10]
1	*	*	0	0	1	M[9]
1	*	*	0	1	0	
1	*	*	0	1	1	
1	*	*	1	0	0	
1	*	*	1	0	1	
1	*	*	1	1	0	
1	*	*	1	1	1	M[17:14]

(a) Truth Table



(b) Logic Cell

Figure 2: Structure of A^2LUT.

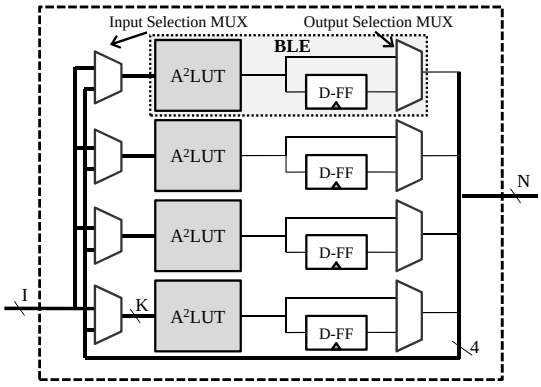


Figure 3: Structure of Logic Cluster Assuming a Size of 4.

$$I = \frac{K}{2} \times (N + 1) \tag{1}$$

In this formula, K is the number of logic cell inputs, and N is the cluster size, i.e., 4. The full connectivity allows each BLE input to be connected to any of the logic cluster inputs and BLE outputs without affecting the connectivity of other BLE inputs – resulting in a set of logically equivalent logic cluster inputs.

4. Performance Evaluation

In order to show the efficiency of our A^2LUT architecture, we evaluate A^2LUT compare with conventional LUT architectures by the largest 12 MCNC benchmark circuits. We implement these circuits by using the cad tools (see Figure 4). This evaluation is performed by using 130nm CMOS technology. The evaluation condition of routing structure and others are shown in Table 4.

Table 3: Mapping Example for A^2LUT.

Appearance Ranking	6-LUT P-representation function binary [hex]	6-LUT Total Coverage [%]	Logic function	Permutation
1	0fff_ffff_ffff_ffff	3.67	$A+B+C+D$	No; use as 4-LUT
2	0000_0000_0000_000f	7.31	$A \cdot B \cdot C \cdot D$	No; use as 4-LUT
3	00ff_ffff_ffff_ffff	10.8	$A+B+C$	No; use as 3-LUT
4	0000_0000_0000_aaaa	14.2	$A \cdot B \cdot \overline{F}$	No; use as 3-LUT
5	0000_0000_8888_8888	17.2	$A \cdot \overline{E} \cdot \overline{F}$	No; use as 3-LUT
6	0000_0000_0000_ffff	19.9	$A \cdot B$	No; use as 2-LUT
7	0000_0000_0000_8888	22.2	$A \cdot B \cdot \overline{E} \cdot \overline{F}$	No; use as 4-LUT
8	0000_ffff_ffff_ffff	24.5	$A+B$	No; use as 2-LUT
9	0000_0000_0000_0080	26.6	$A \cdot B \cdot C \cdot \overline{D} \cdot \overline{E} \cdot \overline{F}$	Yes; swap D for A
10	0000_0000_aaaa_aaaa	28.6	$A \cdot \overline{F}$	No; use as 2-LUT

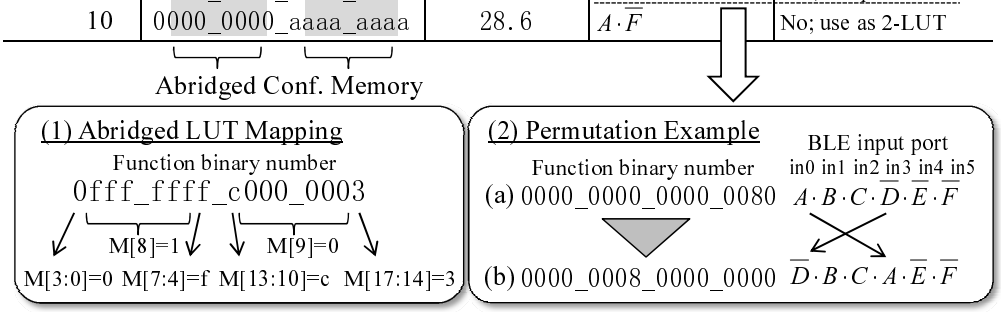


Table 4: Evaluation parameters.

Items	Values
Process technology	130nm CMOS
Chip sizes	$7.4mm \times 7.4mm$
Array size	16×16
Logic element	A^2LUT
Cluster size	4 A^2LUTs in LB
Switch Block type	Wilton ($F_s = 3$)
Connection Block type	normal ($F_c = 0.5$)
# of routing tracks	48/channel
# of single lines	8/channel
# of quad lines	40/channel
# of I/O pins	128
# of conf. bits in all Tile	87,040
# of conf. bits in all IOBs	2,752

Moreover, we developed new technology mapping tool based on EMap [11] algorithm for this evaluation. We added a process to perform matching to the implementable function of A^2LUT from 5- and 6-input function in EMap. The pattern matching of the part of 5- and 6-input functions results in Booleanmatching. This process is based on P-equivalence class. Also, if it's impossible to match in large function, such a 6-input function, it decomposes the function into small functions. At least we can assure mapping to 4-input logic function.

We use T-Vpack [14] and VPR 5.0 [15] in order to get the area and the number of configuration memory bits. Note that we adopt a traditional cluster-based island-style structure as the base of our FPGA architecture, and the cluster size is 4. We also use about the same cluster architecture to evaluate conventional LUTs.

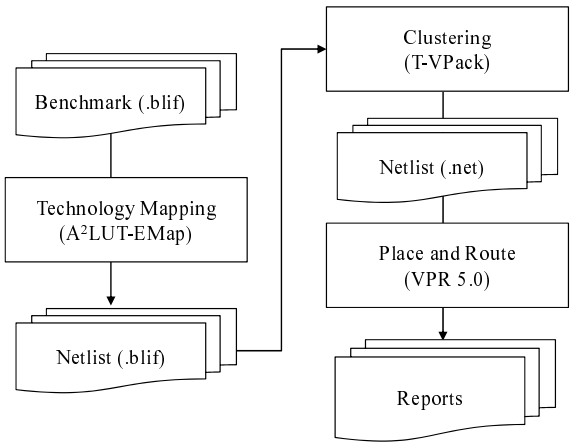


Figure 4: Implementation Flow.

5. Evaluation Results and Remarks

Table 5 shows the implementation results of benchmark circuits for each type of logic cell. Row of BLEs means the number of BLEs after the technology mapping process. Row of LBs means the number of LBs

Table 5: Implementation results.

Circuit name	# of used Logic Cells							
	4-LUT		5-LUT		6-LUT		A^2LUT	
	BLEs	LBs	BLEs	LBs	BLEs	LBs	BLEs	LBs
*alu*4	1,561	403	1,360	345	1,116	293	1,336	345
*apex*2	1,953	506	1,809	464	1,691	428	1,802	474
*apex*4	1,285	339	1,223	317	1,157	293	1,389	384
clma	10,004	2,549	8,862	2,247	7,773	1,952	9,217	2,621
diffeq	1,548	393	1,207	303	954	239	1,320	405
*ex*1010	4,753	1,245	4,315	1,124	3,869	990	4,054	1,154
*ex*5*p*	1,110	290	1,132	290	900	229	1,028	306
*misex*3	1,466	378	1,267	323	1,141	290	1,346	351
pdc	4,940	1,278	4,659	1,191	4,312	1,091	4,858	1,449
*s*298	2,044	520	1,608	404	1,358	342	1,620	441
seq	1,834	473	1,578	401	1,459	367	1,617	424
spla	3,940	1,018	3,417	873	3,171	802	3,774	1,113

after the clustering process.

Figure 5 shows the total number of configuration bits for each individual benchmark. This number is the product of the array size and the number of configuration memory bits per unit tile, which includes a logic block, an SB and a CB. Note that the number of memory bits of the SBs and CBs depends on the channel width. In Fig. 5, the total number of configuration memory for 6-input A^2LUT is approximately 22.1% smaller than that for 6-LUT. Furthermore, the A^2LUT and 5-LUT are about on the same level.

Even if A^2LUT needs a large number of the logic cells, the number of the total configuration memory of A^2LUT is better than that of 6-LUT, for the memory bits per a logic cell are few. Because the quantity of reduction of the configuration memory balances with the increment of the cell for 5-LUT, the quantity of total bits is the almost same. However, the number of the total configuration memory of A^2LUT increased by the additional 2 bits per logic cell for 4-LUT and the large local routing network as same as 6-LUT. Moreover, the number of the logic cells for A^2LUT did not decrease large as compared with 4-LUT.

The result of *alu*4 shows that the total number of configuration memory for the A^2LUT is smaller than that of 4-LUT. It is the reasons that a large number of the BLEs and LBs were able to reduce. On the other hand, the results of *clma*, *diffeq*, *pdc* and *spla* are deterioration in A^2LUT drastically. As for these, the number of the BLEs decreased, but it is a cause that the number of the LBs increases from 4-LUT. This is an issue of the clustering process.

The experimental results are shown in Figure 6; the logic area in 6-input A^2LUT is around 16.8% smaller than that in 6-LUT. However, it is 7.6% larger than that of 5-LUT because the routing area is comparable with that of 6-LUT. This graph shows a tendency same as the configuration memory by the same reason.

Figure 7 shows the critical path delay for each individual benchmark. In this figure, the delay for A^2LUT is 21.9% smaller than that for 6-LUT. In addition, that is 4.0% smaller than that for 5-LUT on an average.

Clearly, our device could be improved significantly as compared with 6-LUT by A^2LUT architecture. Because it has a smaller size of the logic cell than 6-LUT, and the unit routing wire length becomes short, as a result, a routing delay becomes small. In addition, the delay of logic cell oneself is also small.

Several results show that the delay for A^2LUT is smaller than that of 4-LUT. These results vary considerably depending on benchmark circuits. It's depended on the result of technology mapping and clustering process. Namely, it means that A^2LUT has less number of the total logic cells, but cannot necessarily map a circuit with the number of steps of logic cells on critical path as compared with 4-LUT. It's a future work.

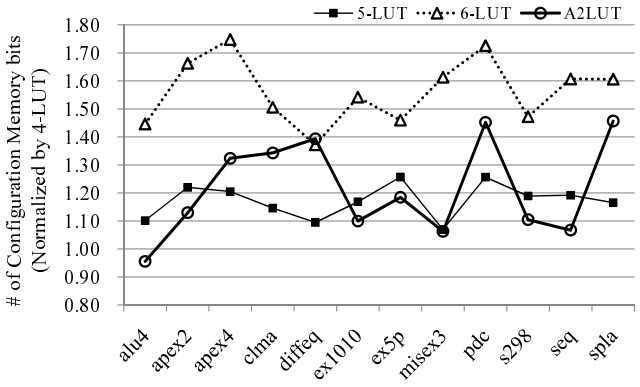


Figure 5: Conf. Memory Evaluation Results.

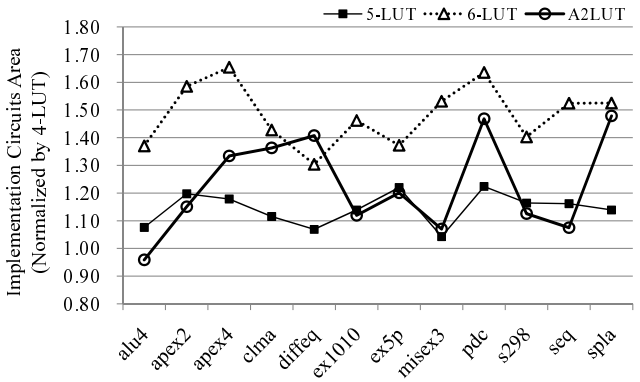


Figure 6: Area Evaluation Results.

6. Conclusion

In this paper, we proposed an abridged configuration memory logic cell named A^2LUT that reduced the chip

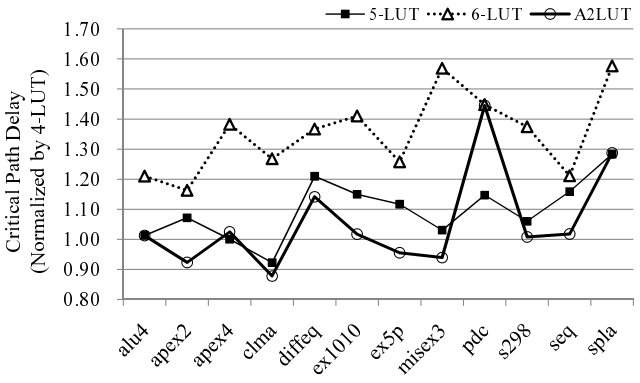


Figure 7: Delay Evaluation Results.

area and the critical path delay. Our approach is to investigate the appearance ratio of the logic functions in a circuit implementation. As well as 4-LUT, the A^2LUT can implement all of 4 input logic functions. As 5 and 6 inputs function, the A^2LUT cannot cover all of P-representative, but it can be implemented the subset of P-representative which include high frequently used.

The experimental results show that the logic area in 6 inputs A^2LUT is 16.8% smaller than that in 6-LUT. Further, the total number of configuration memory bits in the A^2LUT is 22.1% smaller than the number of configuration memory bits in 6-LUT. The critical path delay for the A^2LUT is 21.9% smaller than that for 6-LUT. Moreover, that is 4.0% smaller than that for 5-LUT on an average.

Finally, we developed the prototype chip of A^2LUT architecture. We need to evaluate this chip by using several practical applications. Moreover, our EDA tools for A^2LUT are also required further improvement.

References

[1] Y. Hu, S. Das, S. Trimberger and L. He, “Design and Synthesis of Programmable Logic Block With Mixed LUT and Macrogate,” IEEE TCAD, Vol.28, No.4, pp.591.595, Apl. 2009.

[2] M. Amagasaki, R. Yamaguchi, M. Koga, M. Iida and T. Sueyoshi, “An Embedded Reconfigurable IP Core with Variable Grain Logic Cell Architecture,” International Journal of Reconfigurable Computing, vol.2008, Article ID 180216, 14 pages, 2008. doi:10.1155/2008/180216

[3] Y. Okamoto, Y. Ichinomiya, M. Amagasaki, M. Iida and T. Sueyoshi, “COGRE: A Configuration Memory Reduced Reconfigurable Logic Cell Architecture for Area Minimization,” Proc. of FPL, pp.304-309, Sep. 2010.

[4] M. Iida and T. Sueyoshi, “Proposal and Evaluation of a Logic Block Architecture for Reconfigurable Logic,”Trans. of Information Processing Society of Japan, Vol.43, No.5, pp.1181- 1190, 2002. [in Japanese]

[5] Altera Corporation, “FPGA Architecture White Paper,” WP-01003-1.0, July 2006.

[6] D. Debnath and T. Sasao,“Fast Boolean Matching under Permutation by Efficient Computation of Canonical Form,” IEICE Trans. Fundamentals, Vol. E87-A, No.12, pp.3134-3140, Dec. 2004.

[7] M. A. Harrison, Introduction to Switching and Automata Theory, McGraw-Hill, New York, 1965.

[8] S. Muroga, Logic Design and Switching Theory, John Wiley & Sons, New York, 1979.

[9] M. Shintani, K. Kato, M. Amagasaki, M. Iida and T. Sueyoshi, “An analysis of frequency in the use LUT logic functions based on P-equivalence class,” IEICE technical report 109(198), pp.31-36, 2009. [in Japanese]

[10] J. Cong and Y. Ding, “FlowMap:An optimal technology mapping algorithm for delay optimization in lookup-table based FPGA designs,” IEEE TCAD, Vol 13(1), pp.1-12, Jan. 1994.

[11] J. Lomoureux and S. J. E. Wilton, “On the Interaction between Power-aware FPGA CAD Algorithms,” Proc. of ICCAD, pp. 701-708, Nov. 2003.

[12] V. Betz and J. Rose, “Cluster-Based Logic Blocks for FPGAs: Area- Efficiency vs. Input Sharing and Size,” Proc. of CICC, pp.551.554, 1997.

[13] E. Ahmed and J. Rose, “The Effect of LUT and Cluster Size on Deep-Submicron FPGA Performance and Density,” Proc. of FPGAs, pp.3.12, 2000.

[14] A. Marquardt, V. Bets and J. Rose, “Using Cluster-Based Logic Blocks and Timing-Driven Packing to Improve FPGA Speed and Density,” Proc. of FPGAs, pp.37-46, Feb. 1999.

[15] J. Luu, I. Kuon, P. Jamieson, T. Campbell, A. Ye, M. Fang and J, Rose. “VPR 5.0: FPGA CAD and Architecture Exploration Tools with Single-Driver Routing, heterogeneity and Process Scaling,” Proc. of FPGAs, pp.133-142, Feb. 2009.

[16] H.Yosho, K. Inoue, M. Amagasaki, M. Iida and T. Sueyoshi, “A test scheme for interconnect of FPGA focused on switch block topology,” IEICE technical report 110(362), pp.145-150, 2011. [in Japanese]