PAPER *Special Section on Reconfigurable Systems*

COGRE: A Novel Compact Logic Cell Architecture for Area Minimization

Masahiro IIDA[†a)], **Motoki AMAGASAKI**[†], ***Members***, **Yasuhiro OKAMOTO**[†], **Qian ZHAO**[†], ***Nonmembers***, ***and*** **Toshinori SUEYOSHI**[†], ***Member***

SUMMARY Because of numerous circuit resources of FPGAs, there is a performance gap between FPGAs and ASICs. In this paper, we propose a small-memory logic cell, COGRE, to reduce the FPGA area. Our approach is to investigate the appearance ratio of the logic functions in a circuit implementation. Moreover, we group the logic functions on the basis of the NPN-equivalence class. The results of our investigation show that only small portions of the NPN-equivalence class can cover large portions of the logic functions used to implement circuits. Further, we found that NPN-equivalence classes with a high appearance ratio can be implemented by using a small number of AND gates, OR gates, and NOT gates. On the basis of this analysis, we develop COGRE architectures composed of several NAND gates and programmable inverters. The experimental results show that the logic area of 4-COGRE is smaller than that of 4-LUT and 5-LUT by approximately 35.79% and 54.70%, respectively. The logic area of 8-COGRE is 75.19% less than that of 8-LUT. Further, the total number of configuration memory bits of 4-COGRE is 8.26% less than the number of configuration memory bits of 4-LUT. The total number of configuration memory bits of 8-COGRE is 68.27% less than the number of configuration memory bits of 8-LUT.

key words: *reconfigurable logic, COGRE, NPN-equivalent classes*

1. Introduction

Reconfigurable logic devices (RLDs) provide a low-cost, low-risk implementation medium for increasingly huge and complex systems. These systems are more flexible than application-specific integrated circuits (ASICs) and much faster than general-purpose processors in several application domains [1], [2]. Field programmable gate arrays (FPGAs), which are the most popular RLDs, are used for fine-grain operations and can be configured for implementing various digital circuits. Modern FPGA architectures are typically implemented in cluster-based logic blocks and island-style routing structures, where logic blocks are employed to group basic logic elements (BLEs). The BLEs themselves are groups of flip-flops (FFs) and look-up tables (LUTs). However, for many applications, FPGAs based on LUTs are less efficient than standard cell based designs. This is because FPGAs have numerous circuit resources such as configuration memory bits and selectors that are used to implement any digital circuits.

In order to narrow the performance gap between FPGA and ASIC, many fine-grained logic cells have been proposed as alternative LUTs. An example of such a logic cell is Actel multiplexer-based logic cell that has a simple structure [3]. However, when such logic cells are used together to implement larger input functions, the percentage of functions that cannot be implemented becomes significant. QuickLogic FPGA [4] has 30 inputs and 6 outputs, as well as numerous AND and MUX gates. Although this logic cell offers high logic capacity and minimal logic delay, a large portion of the logic cell is not used. We have studied the architecture of a variable-grain logic cell (VGLC), which has the features of both a coarse-grained and a fine-grained logic cells [5]. A VGLC delivers good performance until the technology mapping phase. Conversely, a routing problem arises owing to the input/output pin overhead in the VGLC. Based on these previous study, we focus on the following two points to overcome these issues.

- ***Logic block 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 the logic area and the number of configuration memory bits in a logic block.
- ***Number of logic block input/output pins***: Generally, the routing area increases with the number of input/output pins in a logic block [6]. As long as an island-style routing structure is employed, it is meaningless if the total number of input/output pins increases more than the number of LUT-based logic cell's input/output pins.

On the basis of these two points, we developed a novel compactly organized generic reconfigurable element (COGRE) logic cell [7]. The key feature of COGRE is its architecture that helps reduce the logic area and the number of configuration memory bits by considering NPN-equivalence [8] functions. An analysis of the synthesized benchmarks indicates that a large percentage of functions in an LUT level netlist are specific NPN-equivalence classes. COGRE can only implement high-appearance-ratio logics, as seen in this analysis. Similar to an *N*-LUT, *N*-COGRE requires at most *N* inputs. Moreover, a conventional CAD tool set can still be used for COGRE with minor modifications. In this paper, we extended our work in [7] to further reduce the number of configuration memory bits and area by applying a cluster-based logic block. We also compare the efficiency and effectiveness of our logic cell with those of the LUT architectures.

Manuscript received April 22, 2011.
Manuscript revised September 5, 2011.

[†]The authors are with the Graduate School of Science and Technology, Kumamoto University, Kumamoto-shi, 860–8555 Japan.
a) E-mail: iida@cs.kumamoto-u.ac.jp

DOI: 10.1587/transinf.E95.D.294

The rest of the paper is organized as follows. Section 2 discusses the COGRE architecture and CAD flow. Section 3 describes the evaluation process, and Sect. 4 presents a discussion on the results obtained herein. Section 5 describes related research works. Finally, Sect. 6 includes the conclusions of this study and provides an overview of our future research plan.

2. Logic Cell Design

To design small and efficient logic cells, we must know the nature of the logic functions used for circuit implementation. In this section, we report the investigation of the appearance ratio of logic functions in circuit implementation. We then propose a logic cell architecture for area minimization.

2.1 LUT-Based FPGA

An SRAM-based FPGA is typically a two-dimensional array of configurable logic blocks (LBs) that are interconnected by connection modules, wires, and a switch module. An LB comprises one or more BLEs, where the truth tables of the implemented functions are stored in SRAM tables. The structure of a cluster-based LB, which is used in our work, is shown in Fig. 1 (a). An LB comprises external inputs, clocks, outputs, and BLEs. The multiplexers in the LB provide arbitrary connections from the external inputs of the LB and the outputs of the BLEs to the inputs of any BLE. Additionally, the structure of a BLE is shown in Fig. 1 (b). It consists of a three-input LUT, a D flip-flop, and a 2:1 MUX that enables the transfer of the data to the ouput of the BLE, either from the flip-flop or directly from the LUT. Previously, the internal connections in a cluster LB were assumed to be fully connected, where each BLE input could come from any cluster input or feedback connection. An N-input LUT can generally implement any N-input logic function. However, 2^N configuration memory bits and 2^N-1 MUXes are required for an N-input LUT.

2.2 NPN-Equivalence Class

Before discussing the appearance ratio of logic functions, we must explain an important concept, the NPN-equivalence class [8], for classifying the logic functions.

***Definition (NPN-Equivalence)*:** Given two functions F and G, if function F can be derived from G by negating (N) and/or permuting (P) some inputs and/or by negating (N) the output, functions F and G are NPN equivalent and belong to the same NPN-equivalence class.

We consider the logic functions shown in Fig. 2, for example. Given a function $Y1 = A + BC$, if we permute A and B, we obtain a function $Y2 = B + AC$. If we negate the some or all inputs of $Y1$, we obtain functions such as $Y3$ and $Y4$. If the output is negated, we can obtain $Y5$. Finally, $Y6 = \overline{\overline{B} + \overline{AC}}$ is obtained by negating all inputs, permuting inputs A and B, and negating the output of $Y1$. In other words,

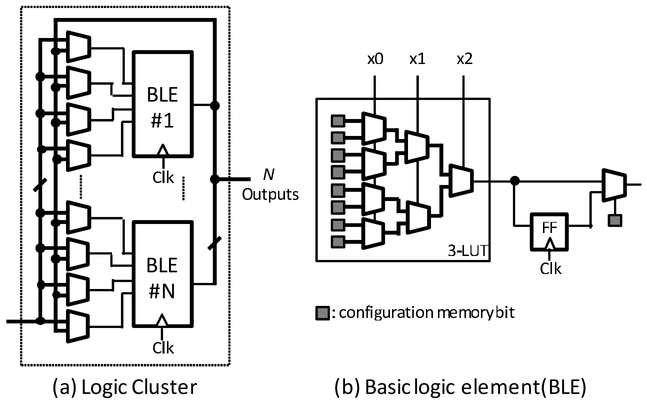
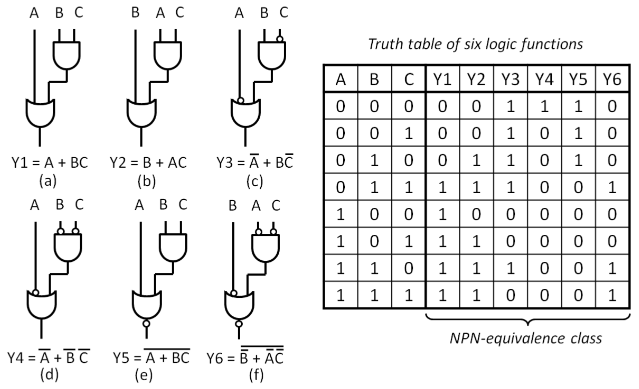


Fig. 1 Structure of (a) Logic block and (b) Basic Logic element.


A B C
B A C
A B C
Y1 = A + BC (a)
Y2 = B + AC (b)
Y3 = $\overline{A}$ + B$\overline{C}$ (c)
A B C
A B C
B A C
Y4 = $\overline{A}$ + $\overline{B}$ $\overline{C}$ (d)
Y5 = $\overline{A + BC}$ (e)
Y6 = $\overline{\overline{B} + \overline{AC}}$ (f)
Truth table of six logic functions
NPN-equivalence class

A	B	C	Y1	Y2	Y3	Y4	Y5	Y6
0	0	0	0	0	1	1	1	0
0	0	1	0	0	1	0	1	0
0	1	0	0	1	1	0	1	0
0	1	1	1	1	1	0	0	1
1	0	0	1	0	0	0	0	0
1	0	1	1	1	0	0	0	0
1	1	0	1	1	1	0	0	1
1	1	1	1	1	0	0	0	1

Fig. 2 Example of NPN-equivalence Class: (a) Original function; (b) Input permutation; (c) Input negation; (d) All inputs negation; (e) Output negation; (f) Inputs negation, inputs permutation, and output negation.

functions $Y1$-$Y6$ are NPN equivalent. It should be noted that for the above mentioned example functions, a total of 48 logic functions belong to the same NPN-equivalence class. Generally, there are at most $2^{(N+1)} \cdot N!$ distinct functions that belong to the same NPN-equivalence class for an N-input function. In this expression, $2^{(N+1)}$ indicates the negation of the inputs and the output, and $N!$ indicates the permutation of the inputs.

2.3 Appearance Ratio of Logic Functions

To determine the appearance ratio of the logic functions in a circuit implementation, we perform technology mapping of several benchmark circuits and count the logic functions that are implemented as LUTs. In this investigation, mapping is performed using a priority-cut-based LUT mapper [9]. This mapping tool is implemented on an ABC platform developed by the University of California, Berkeley (UCB) [10]. Moreover, we prepare 120 MCNC circuits [11] as our benchmark.

Table 1 lists the appearance ratios of the logic functions obtained by the 4-LUT-based technology mapping. These logic functions are ranked in the descending order of their appearance ratios. In this table, the input variables of the

Table 1 Appearance ratio of 4-input logic functions.

Rank	Logic Function	Appearance Ratio [%]
1	$ABCD$	40.0
2	$AB(C + D)$	13.5
3	$A(B + C + D)$	10.5
4	$A(B + CD)$	9.6
5	$A + BC + \overline{B}D$	6.0
6	$AB + CD$	5.4
7	$AB + \overline{A}CD$	2.7
-	Others	12.3

logic functions are represented by A, B, C, and D. It should be noted that we group logic functions that belong to the same NPN-equivalence class. This is because these logic functions are supposed to have a similar structure.

The total number of NPN-equivalence classes of 4-input logic functions is 222. As shown in Table 1, the top 7 NPN-equivalence functions cover 87.7% of the logic functions implemented by 4-LUTs. In other words, 87.7% of the 4-input logic functions can be implemented by using logic cells that cover only 7 out of the 222 NPN-equivalence classes. By effectively using the abovementioned appearance ratios of the logic functions, we can design highly efficient logic cells.

2.4 Proposed 5-Input Logic Cell

First, we find that implementing the top 7 logic functions listed in Table 1 needs a maximum of four 2-input AND and/or OR gates and one NOT gate. Moreover, because AND gates and OR gates belong to the same NPN-equivalence class, we can interchange these 2-input logic gates by inverting their inputs and an output.

On the basis of this assumption, we propose a 5-input logic cell, namely 5-COGRE, as shown in Fig. 3. The COGRE stands for compactly organized generic reconfigurable element. In this figure, the NOT gates indicate programmable inverters that are inverting or non-inverting buffers depending on the configuration memory bit. Actually, when the output of one COGRE cell is connected to another COGRE's input, the output programmable inverter can be redundant. Hence, the output programmable inverter does not real exist. And for the adjustment of primary output signals, we place programmable inverters in I/O pads instead. By using 5-COGRE, we can implement the top 7 logic functions listed in Table 1. It is found that, this 5-COGRE can cover 93.4% of the logic functions in benchmarks implemented as LUTs (for 4 or less input functions). Since 5-COGRE can implement any 2-input logic function, more than 2-input functions that cannot be covered are divided into 2-input logic and then mapped to COGRE. However, the COGRE architecture cannot implement full adder for arithmetic circuits efficiently like LUT in commercial FPGAs, because 3-XOR is needed while the COGRE cannot provide. We are planning to improve this in the future work.

Generally, a conventional 4-LUT requires 16 configuration memory bits for implementing 4-input logic functions. On the other hand, 5-COGRE requires only 8 configuration bits. In other words, the number of configuration bits required by 5-COGRE is half of that required by 4-LUT. In addition, 5-COGRE can implement several 5-input logic functions. These 5-input logic functions are very useful owing to the fact that they have a relatively high appearance ratio.

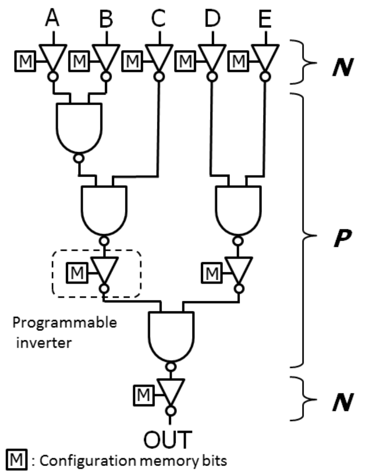


Fig. 3 Proposed 5-COGRE logic cell.

Because 5-COGRE has five input pins, we require special handling methods to implement 4-input logic functions, such as bridging or constant assignment (e.g, H-level or L-level logic) of input pins [12]. To carry out constant assignment, we require both the VDD and the GND signals. However, we can omit either the VDD signal or the GND signal because we can generate the omitted signal by using programmable inverters connected to the input pins. Clearly, these operations can be used to implement 3 or a lesser number of input logic functions.

2.5 COGRE Class

On the basis of the concept of 5-COGRE, to adapt to various input logic functions, we proposed an input extension model of the COGRE permutable logic part. An N-input COGRE structure is called N-COGRE, and all COGRE logic cells are named the COGRE class. It is important to note that our method of designing logic cells with a tree structure is applicable to logic functions with different input sizes. We simply need to vary the levels of the tree of programmable inverters and NAND gates to design different logic cells. In order to obtain a minimum critical path delay, we have to design N-COGRE in least logic levels. Therefore, the logic depth of the N-input logic is $\lceil logN \rceil$. Examples of 4- and 6- to 8-input COGREs are shown in Fig. 4.

We also investigated the appearance ratio of 6-input logic functions in the same manner as we did for 5-COGRE. The data shows a similar result; in other words, small portions of NPN-equivalence class can cover large portions of the logic functions that are used to implement circuits. How-

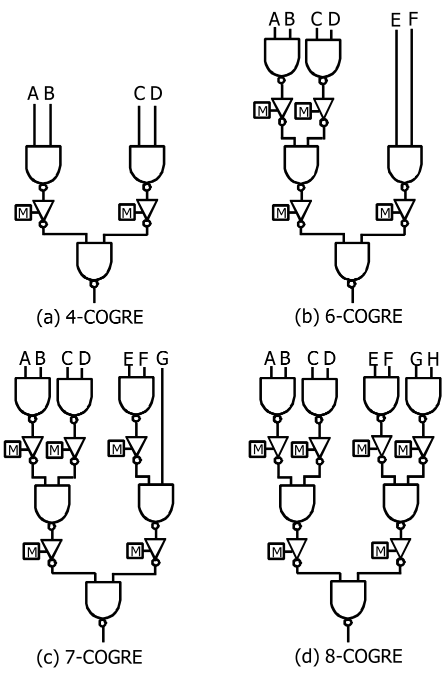


Fig. 4 Other COGRE permutable logics.

ever, as the number of inputs increases, the logic functions grow exponentially. Therefore, in this research, we only explore the 8-input COGRE at most.

2.6 CAD Tools for Proposed Logic Cells

To implement various applications into our COGRE architecture, we require several CAD tools. It should be noted that we can also use the existing CAD tools after minor modifications. However, unlike LUT-based FPGAs, COGREs require special technology mapping tools because COGREs can implement only a subset of logic functions. For this reason, we have developed a library-based technology mapping tool, as in [13]. This technology mapping tool is based on a priority-cut-based LUT mapper [9]. We have created a library that includes implementable NPN-equivalence class representatives for each COGRE. This library is required for determining whether a given target function is implementable or not. In addition, we use an efficient boolean matcher [14] to generate NPN-equivalence class representatives.

3. Evaluation Method

We evaluate 4 types of LUTs and 4 types of COGREs to show the efficiency of our architecture. In particular, we evaluate the area, the total number of routing tracks, the total number of configuration memory bits, and the critical path delay. In this section, we first describe the evaluation flow and then introduce the architectural model.

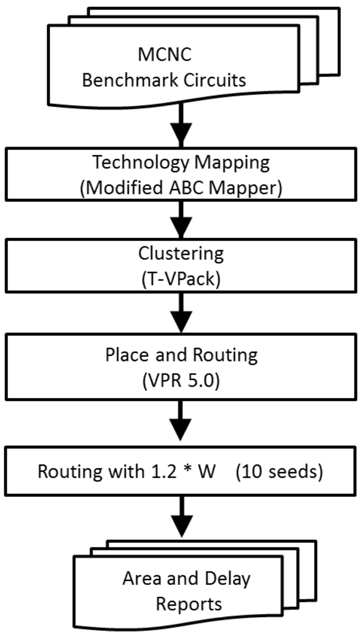


Fig. 5 Evaluation flow.

Table 2 20 largest MCNC benchmark circuits.

alu4	apex2	apex4	bigkey	clma
des	diffeq	dsip	elliptic	ex1010
ex5p	frisc	misex3	pdc	s298
s38417	s38584.1	seq	spla	tseng

3.1 Evaluation Flow

Figure 5 shows the evaluation flow. This flow is typically a CAD flow for circuit implementation on a cluster-based island-style FPGA. We use the 20 largest MCNC circuits listed in Table 2 as the benchmark set. The details of our modified technology mapping tool are described in Sect. 2.6. Further, we use the original T-VPack [15] to perform clustering and VPR 5.0 [16] to place and route circuits. Finally, we evaluate the target architectures by analyzing the reports derived from VPR.

Some points concerning the use of VPR should be explained here. We perform VPR multiple times for each circuit. After the first time, we obtain an initial channel width on the basis of which we change the delay of the connection box (CB) in the architecture file. The structure of the CB is determined on the basis of the channel width and the inputs of the LB. We obtain the delay of the CB by synthesizing it with a standard cell library. After the second time, we obtain a reasonable channel width which we multiply by 1.2 to obtain the channel width of the final VPR process. Again, the delay of the CB in the architecture file is changed. In the final VPR process, we perform placement by using 10 different VPR seeds for each benchmark circuit. The final result is obtained by taking the average of the 10 results.

As we mentioned in Sect. 2.4, because we have saved the output programmable inverter, we have to add an output

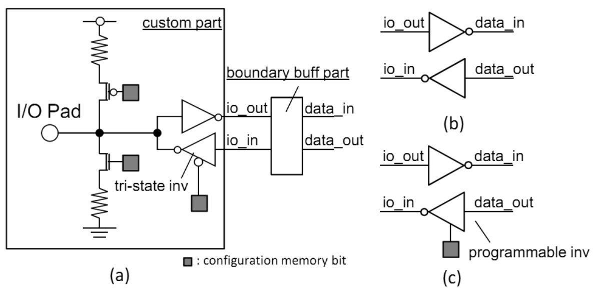


Fig. 6 I/O block (a) I/O block overview (b) Boundary buffer part for LUT (c) Boundary buffer part for COGRE.

programmable inverter to each I/O block. For both COGRE and LUT device architecture, a boundary buffer part is attached to each I/O block. Figure 6 (a) shows the structure of I/O block and boundary buffer part. Figure 6 (b) and (c) show boundary buffer parts for LUT and COGRE architecture, respectively. The data_in and data_out signals are connected to routing tracks of reconfigrable logic array.

3.2 Logic Block Structure

Conventional FPGAs adopt a cluster-based architecture to improve performance [17]. By packing several near logics into single LB, the local connections within a cluster are much faster than global routings. In this evaluation, we assume the cluster architecture with size of 4, which are adopted in several conventional FPGAs. Figure 7 shows the cluster structure. One BLE consists of K-input COGREs, D-FFs, and an output-selection MUX. The number of cluster inputs I is determined by $I = \frac{K}{2} \times (N + 1)$ [18]. Where, K is the number of logic cell inputs, and N is the cluster size. Several input-selection MUXes select the cluster inputs, cluster outputs, or the GND signal to the input pins of the COGRE BLE. It should be noted that this GND signal is required for the constant assignment of the COGRE input pins so that some relative logic functions can be implemented.

The LUT architectures are also based on a similar cluster structure. The only difference is that the GND signal for the input-selection MUXes is not required. Because LUTs can implement any logic function, the constant assignment of input pins is not necessary.

3.3 Area and Delay Model

We synthesize both COGREs and LUTs using Synopsys Design Compiler with e-Shuttle 1.2 V, 65 nm CMOS technology standard cell library. The physical parameters of area, delay, and the number of configuration memory bits of all target architectures are listed Table 3. These parameters are used in the FPGA architecture description file in VPR. For the routing architecture, we use wilton style SB, whose Fs is set to 3. The value of Fc of CB is set to 0.5. The values of R and C of routing wire segment are calculated from the area of one tile. It should be noted that CBs and switch

Table 3 Summary of area, delay, and number of configuration memory bits.

	BLE			Logic Cluster (including 4 BLEs)		
	Area [μm^2]	Delay [ps]	# of conf. bits	Area [μm^2]	Delay [ps]	# of conf. bits
4-LUT	377.76	127.14	17	2,778.24	230.05	132
5-LUT	759.84	149.75	33	4,690.56	265.22	212
6-LUT	1,392.48	167.08	65	7,977.60	300.22	380
7-LUT	2,265.12	195.64	129	12,044.16	345.42	656
8-LUT	4,471.20	236.36	257	21,555.84	373.71	1,188
4-COGRE	103.68	116.01	7	1,681.92	218.92	92
5-COGRE	161.28	132.42	9	2,449.92	257.91	136
6-COGRE	216.00	137.07	11	3,271.68	270.21	164
7-COGRE	264.00	136.67	13	4,039.68	286.45	192
8-COGRE	314.40	137.36	15	4,928.64	274.71	220

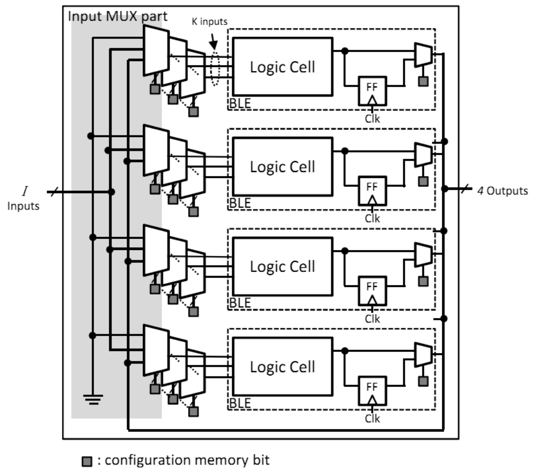


Fig. 7 Structure of logic cluster comprising 4 BLEs.

boxes (SBs) for the routing structures are also synthesized under the same condition. We do not show the values here because they vary with the channel width of different circuits. In the I/O blocks, we only consider the area and delay of the buffer part. The elimination of the custom part which includes analog circuits does not affect our evaluation, because the structure of the custom part is the same for both LUT and COGRE,

The first part of Table 3 corresponds to the model of BLE. Each BLE includes a D-FF, an output-selection MUX, and all the configuration memory bits. It is important to note that output-selection MUX uses one bit of memory. Moreover, all the configuration memory bits are evaluated with D-latches. The delay shown in this table is a combinational delay from the input to the output of the BLE.

The second part of Table 3 includes the model of a cluster, as shown in Fig. 7. This model includes 4 BLEs, input-selection MUXes, and the configuration bits for MUX control. The delay is the logic delay from the input of a cluster to the output of the output-selection MUX.

We speculate that the actual FPGAs are designed using pass-transistor MUXes and that the configuration memory bits are the manufactured SRAM cells. However, detailed constitution and area are not opened. Although [19] suggests the use of the LB area, the present study includes a routing region that has a large number of tracks. For this reason, we believe that a fair comparison can be made by using the standard cell information.

4. Experimental Results

4.1 Implementation Area

Figure 8 shows the logic area evaluation for each individual benchmark. The data is normalized by the performance of 4-LUT. In this figure, the logic area of 4-COGRE is smaller than that of 4-LUT and 5-LUT by approximately 35.79% and 54.70%, respectively. The logic area of 8-COGRE is approximately 75.19% smaller than that of 8-LUT. For applications that require better area efficiency, the 5-COGRE performs the best.

Figure 9 shows the total number of routing tracks for each individual benchmark. The data is normalized by the performance of 4-LUT. As seen in this figure, the total number of routing tracks of 4-COGRE exceeds that of 4-LUT and 5-LUT by 3.71% and 9.65%, respectively. Further, the total number of routing tracks for 8-COGRE is almost 11.72% greater than that for 8-LUT. These results show similar findings to those of the routing area evaluation.

Figure 10 shows the total area for each individual benchmark. The data is normalized by the performance of 4-LUT. The total area includes the logic area and the routing area. These areas are calculated by multiplying the logic or the routing area of one tile by *array size* × *array size*, which are introduced in the previous section. In this figure, the total area of 4-COGRE is smaller than that of 4-LUT and 5-LUT by approximately 17.55% and 30.38%, respectively. The total area of 8-COGRE is approximately 67.56% smaller than that of 8-LUT. We can see that although the number of routing tracks of COGREs is greater than that of LUTs, which will lead to a larger routing area, the total area of COGREs is still smaller than that of LUTs for better logic area performance. The logic area of N-LUT is $O(2^N)$. On the other hand, the logic area of N-COGRE is only $O(N)$. This means that when the number of inputs increase, the logic area of LUTs increases exponentially while that of COGREs increases linearly. Therefore, the logic area advantage of the proposed COGRE architecture is more significant when implementing larger input logic cells.

4.2 Total Number of Configuration Memory Bits

Figure 11 shows the total number of configuration memory bits for each individual benchmark. The data is normalized by the performance of 4-LUT. 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. It should be noted that the number of memory bits of the SBs and CBs depends on the channel width. In this evaluation, we assume that in SBs and CBs are built with standard cell MUXes, and not two-level MUXes [20] that are used in VPR, because our area evaluation is based on the standard cell library.

As shown in Fig. 11, the total number of configuration

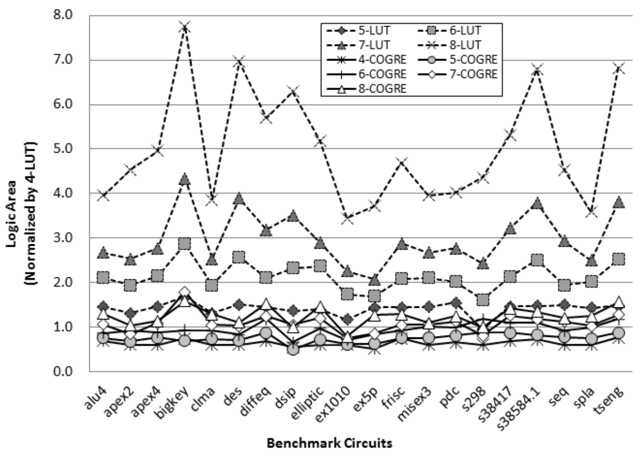


Fig. 8 Logic area.

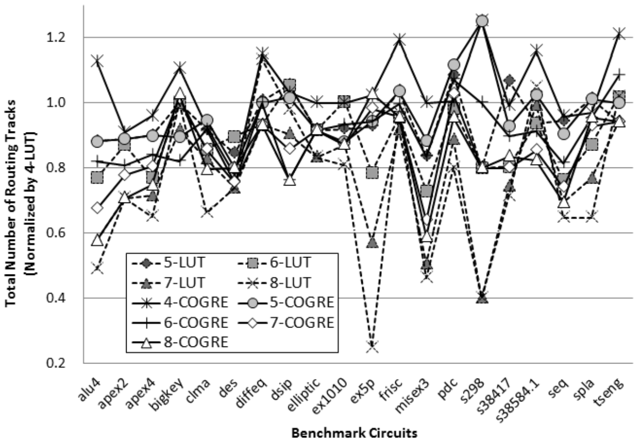


Fig. 9 Total number of routing tracks.

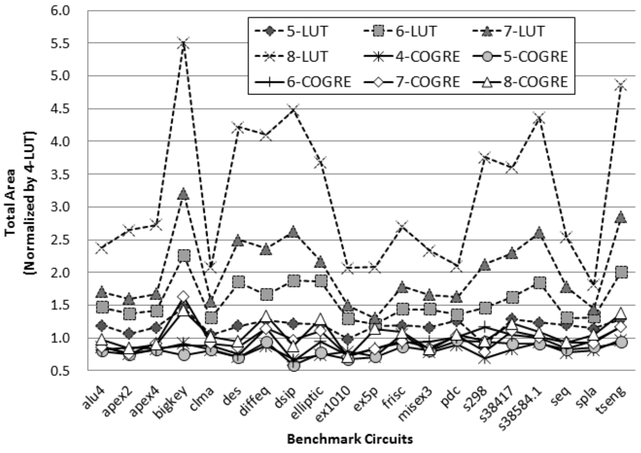


Fig. 10 Total area.

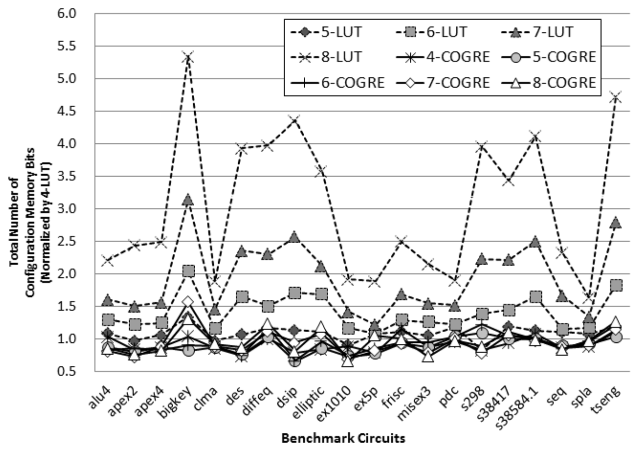


Fig. 11 Total number of configuration memory bits.

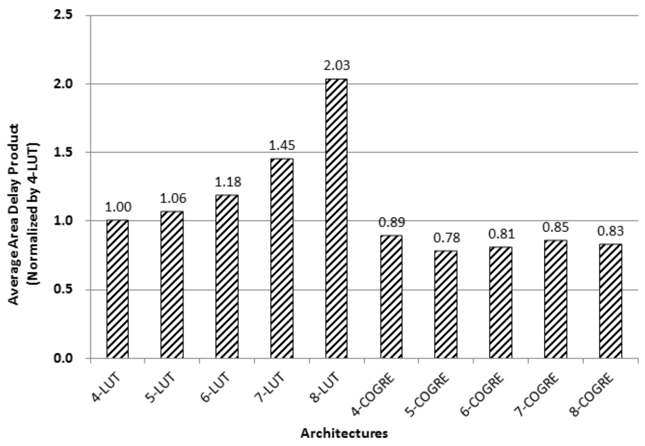


Fig. 13 Average area delay product.

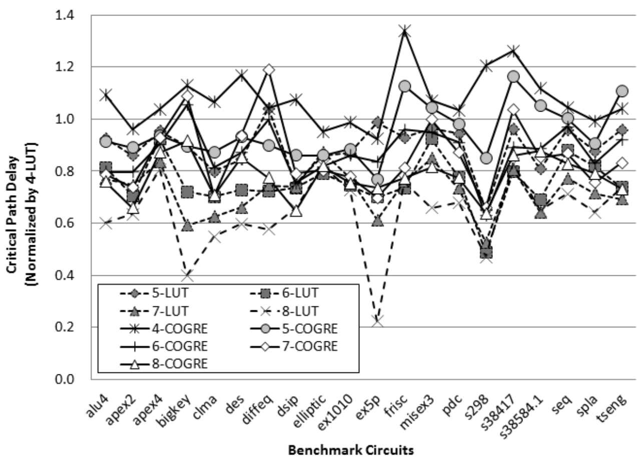


Fig. 12 Critical path delay.

bits of 4-COGRE is approximately 8.26% and 15.61% less than that of 4-LUT and 5-LUT, respectively. Moreover, the total number of configuration memory bits for 8-COGRE is approximately 68.27% less than that for 8-LUT. We can clearly see that COGREs use less configuration memory bits than LUTs. For applications that require less configuration bits, the 5-COGRE performs the best on average. However, several results show that the total number of configuration memory bits for 8-COGRE is less than that of 5-COGRE. Similar to the logic area performance, the configuration memory bits of the logic cell of N-LUT are $O(2^N)$, those of the logic cell of N-COGRE are only $O(N)$. More configuration memory bits can be saved by using COGRE when implementing larger input logic cells.

4.3 Critical Path Delay

Figure 12 shows the critical path delay for each individual benchmark. As shown in this figure, the delay for 8-COGRE is approximately 22.44% greater than that for 8-LUT. However, the delay for 8-COGRE is approximately 22.02% less than that for 4-LUT. Further, the delay in 4-COGRE is approximately 7.71% greater than that in 4-LUT and 20.65% greater than that in 5-LUT. These results vary considerably depending on the benchmark circuits. For applications that require better speed efficiency, 8-COGRE performs the best in the COGREs.

4.4 Area Delay Product

Figure 13 shows the average area delay product for each evaluated target architecture. The data is normalized by the performance of 4-LUT. In this figure, the area delay product of 4-COGRE is smaller than that of 4-LUT and 5-LUT by approximately 10.82% and 16.09%, respectively. The area delay product for 8-COGRE is approximately 59.29% smaller than that for 8-LUT. For applications that require better balance between area and delay, the 5-COGRE performs the best. The results show that each COGRE architecture has better area delay product performance than LUT with the same number of inputs. Moreover, the area delay product of LUTs decreases quickly with an increase in input N. On the other hand, COGREs have better structure balance so their area delay product does not change much from 4- to 8-COGRE.

The area delay product shows the performance based on the consideration of both the area and the delay. We can see that although the speed of COGREs is slower than that of LUTs on an average, the area delay product shows significant improvement because of COGRE's better area performance.

5. Related Studies

To reduce the circuit resources, several researchers have attempted to design logic blocks that are more compact than conventional LUTs. Kimura *et al.* [21] proposed a folding method of logic functions for reducing the number of configuration memory bits. This architecture will be used to implement arithmetic functions, rather than a more general logic circuit. Other researchers have designed compact logic blocks on the basis of the concept of NPN equivalence [14].

Meyer and Kocan [1] proposed a novel logic block architecture, in which the configuration SRAMs of LUTs are shared among NPN-equivalent functions to reduce the number of configuration memory bits. Because the structures of these logic blocks are constructed on the basis of conventional LUTs, significant improvement was not achieved. [12], [22], and [23] include methods for designing new universal logic modules (ULMs), which use the concept of NPN-equivalence. In [23], the proposed 4-input ULM is composed of only 13 configuration memories. However, these logic cells described only four inputs and did not evaluate the result of placement and routing. In [24], Hu *et al.* proposed a heterogeneous programmable logic block using a combination of LUTs and macrogates. They presented a method for extracting a small set of logic functions that can implement large portions for the given FPGA applications. By using this report, they developed the macro-gates that could cover many logic functions used in the circuit implementation. However, they did not focus on the final placement and routing results.

6. Conclusion

In this paper, we propose a small-memory logic cell named COGRE that helps minimize the chip area. Our approach is to investigate the appearance ratio of the logic functions in a circuit implementation. Moreover, we group the logic functions on the basis of the NPN-equivalence class. The results of our investigation show that only small portions of the NPN-equivalence class can cover large portions of the logic functions used to implement circuits. Further, we found that NPN-equivalence classes with a high appearance ratio can be implemented by using a small number of AND gates, OR gates, and NOT gates. On the basis of this observation, we develop *N*-COGRE architectures composed of several NAND gates and programmable inverters.

The experimental results show that the logic area of 4-COGRE is smaller than that of 4-LUT and 5-LUT by approximately 35.79% and 54.70%, respectively. The logic area of 8-COGRE is 75.19% less than that of 8-LUT. Further, the total number of configuration memory bits of 4-COGRE is 8.26% less than the number of configuration memory bits of 4-LUT. The total number of configuration memory bits of 8-COGRE is 68.27% less than the number of configuration memory bits of 8-LUT.

In the future, we will investigate the effectiveness of our cluster architecture. Because each n-COGRE cannot implement all logic functions, the logic depth evaluation showed a worse result than that of LUT-based logic cell. For example when implementing a full adder, more logic cells are needed by COGRE than LUT. Actually, the COGRE performs not much efficiently for arithmetic applications yet. To overcome this disadvantage, we will introduce a cluster structure with a combination of COGREs and LUTs. In addition, we will study the heterogeneous COGRE cluster.

References

[1] J. Meyer and F. Kocan, "Sharing of SRAM tables among NPN-equivalent LUTs in SRAM-based FPGAs," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.15, no.2, pp.182–195, Feb. 2007.

[2] T. Sueyoshi and M. Iida, "Configurable and reconfigurable computing for digital signal processing," IEICE Trans. Fundamentals, vol.E85-A, no.3, pp.591–599, March 2002.

[3] Actel, Application Note: Introduction to Actel FPGA Architecture, 1997.

[4] QuickLogic Corp., Eclipse II Family Data Sheet, 2004.

[5] 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, p.14, 2008.

[6] J. He and J. Rose, "Advantages of heterogeneous logic block architecture for FPGAs," Proc. CICC, pp.7.4.1–7.4.5, 1993.

[7] Y. Okamoto, Y. Ichinomiya, M. Amagasaki, M. Iida, and T. Sueyoshi, "COGRE: A configuration memory reduced reconfigurable logic Cell Architecture for Area Minimization," Proc. FPL, pp.304–309, Aug. 2010.

[8] 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.

[9] A. Mishchenko, S. Cho, S. Chatterjee, and R. Brayton, "Combinational and sequential mapping with priority cuts," Proc. ICCAD, pp.354–361, 2007.

[10] R. Brayton and A. Mishchenko, "ABC: An academic industrial-strength verification tool," Proc. CAV10, LNCS 6174, pp.24–40, Springer, 2010.

[11] K. McElvain, "IWLS'93 Benchmark Set: Version 4.0," Distributed as part of the MCNC international workshop on logic synthesis '93 benchmark distribution, May 1993.

[12] C. Lin, M. Marek-Sadowska, and D. Gatlin, "Universal logic gate for FPGA design," Proc. ICCAD, pp.164–168, 1994.

[13] A. Kennings, K. Vorwerk, A. Kundu, V. Pevzner, and A. Fox, "FPGA technology mapping with encoded libraries and staged priority cuts," Proc. FPGAs, pp.143–150, 2009.

[14] D. Chai and A. Kuehlmann, "Building a better boolean matcher and symmetry detector," Proc. DATE, pp.1079–1084, 2006.

[15] A. Marquardt, V. Bets, and J. Rose, "Using cluster-based logic blocks and timing-driven packing to improve FPGA speed and density," Proc. FPGAs, pp.37–46, 1999.

[16] 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. FPGAs, pp.133–142, 2009.

[17] V. Betz and J. Rose, "Cluster-based logic blocks for FPGAs: Area-efficiency vs. input sharing and size," Proc. CICC, pp.551–554, 1997.

[18] E. Ahmed and J. Rose, "The effect of LUT and cluster size on deep-submicron FPGA performance and density," Proc. FPGAs, pp.3–12, 2000.

[19] C.H. Ho, P.H.W. Leong, W. Luk, S.J.E. Wilton, and S. Lopez-Buedo, "Virtual embedded blocks: A methodology for evaluating embedded elements in FPGAs," Proc. FCCM, pp.35–44, 2006.

[20] D. Lewis, E. Ahmed, G. Baeckler, V. Betz, M. Bourgeault, D. Cashman, D. Galloway, M. Hutton, C. Lane, A. Lee, P. Leventis, S. Marquardt, C. McClintock, K. Padalia, B. Pedersen, G. Powell, B. Ratchev, S. Reddy, J. Schleicher, K. Stevens, R. Yuan, R. Cliff, and J. Rose, "The Stratix II logic and routing architecture," Proc. FPGAs, pp.14–20, 2005.

[21] S. Kimura, T. Horiyama, M. Nakanishi, and H. Kajihara, "Folding of logic functions and its application to look up table compaction," Proc. ICCAD, pp.694–697, 2002.

[22] S. Thakur and D.F. Wong, "On designing ULM-based FPGA logic

modules," Proc. FPGAs, pp.3–9, 1995.

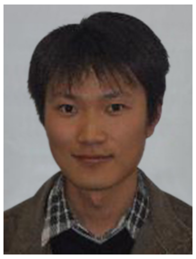
[23] Z. Zilic and Z.G. Vranesic, "Using decision diagrams to design ULMs for FPGAs," IEEE Trans. Comput., vol.47, no.9, pp.971–982, Sept. 1998.

[24] Y. Hu, S. Das, S. Trimberger, and L. He, "Design and synthesis of programmable logic block with mixed LUT and macrogate," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.28, no.4, pp.591–595, April 2009.

Masahiro Iida received his B.E. degree in Electronic Engineering from Tokyo Denki University in 1988. He was a research engineer at Mitsubishi Electric Engineering Co., Ltd. from 1988 to 2003. He received his M.E. degree in Computer Science from Kyushu Institute of Technology in 1997. Further, he received his D.E. degree from Kumamoto University, Japan, in 2002. He has been an associate professor in the Department of Computer Science at Kumamoto University since February 2003. During 2002–2005, he held an additional post as a researcher at PRESTO, Japan Science and Technology Corporation (JST). His current research interests include high-performance low-power computer architectures, reconfigurable computing systems, VLSI devices and design methodology. He is amember of the IPSJ.

Motoki Amagasaki received the B.E. and M.E., degrees in Control Engineering and Science from Kyushu Institute of Technology, Japan in 2000, 2002, respectively. He was a design engineer at NEC Micro Systems Co., Ltd. from 2002–2005. He received the D.E. degree from Kumamoto University, Japan, in 2007. He has been an assistant professor in the Department of Computer Science at Kumamoto University since October 2007. His research interests reconfigurable system and VLSI design. He is a member of IPSJ and IEEE Computer Society.

Yasuhiro Okamoto received the B.E. degree in Computer Science from Kumamoto University in 2009. Further, he received the M.E. degree in Computer Science and Electrical Engineering from Kumamoto University in 2011. He joined Pankaku Inc. in 2011.



Qian Zhao received the B.E. degree in Control Engineering and Science from Qingdao University of Science and Technology, China, in 2007. Further, he received the M.E. degree in Computer Science and Electrical Engineering from Kumamoto University in 2011. He is now a doctoral student at Kumamoto University. His research interest is reconfigurable device architecture.

Toshinori Sueyoshi received the B.E. and M.E. degrees in Computer Science and Communication Engineering from Kyushu University in 1976 and 1978 respectively. From 1978 to 1987, he was a research associate at Kyushu University, where he received D.E. degree in 1986. From 1987 to 1989, he was an associate professor of Information System at Kyushu University. From 1989 to 1997, he was an associate professor of Artificial Intelligence at Kyushu Institute of Technology. Since 1997 he has been a professor of Computer Science at Kumamoto University. He is also a guest professor at Kyushu University. His primary research interests include computer architecture, reconfigurable systems, parallel processing. He served as Chair of the Technical Committee on Reconfigurable Systems of the IEICE, Chair of the Technical Committee on Computer Systems of the IEICE, Chair of the IEEE Computer Society Fukuoka Chapter, and Director of the IPSJ Kyushu Chapter. Currently he also serves as Vice Chair of the IEEE CAS Society Fukuoka Chapter and Chief Director of the FPGA Consortium Japan.