A Proposal of Programmable Logic Architecture for Reconfigurable Computing

Masahiro Iida[1]* and Toshinori Sueyoshi[2]
[1]Graduate School of Science and Technology, Kumamoto University
[2]Department of Computer Science, Faculty of Engineering
Kumamoto University, Kumamoto, Japan, E-mail: sueyoshi@cs.kumamoto-u.ac.jp

Abstract:
Reconfigurable computing is a new computing paradigm which has more potential in terms of performance and flexibility. Reconfigurable computing systems are opening a new era in digital signal processing such as multimedia, communication and consumer electronics because they can filter data rapidly and excel at pattern recognition, image processing and encryption. Although many reconfigurable computing systems use a conventional programmable device, they carry several serious problems to be solved. This paper proposes a logic block architecture of programmable device suitable for the reconfigurable computing. Compared to conventional logic blocks, our logic block can improve implementation density, efficiency and speed.

1. Introduction

Programmable logic devices have produced a new computing paradigm, which is called reconfigurable computing, because they have given the flexibility of hardware to applications. A system that uses the reconfigurable computing technology obtain not only flexibility, but also high performance and low power consumption. However, a serious problem for reconfigurable computing is that conventional FPGAs reconfigure slowly and do not have enough logic capacity. Consequently, it has been necessary to develop a new device accordingly.

In this paper, we focus on the programmable logic architecture, and we propose a novel logic block. Our logic block contains configuration data cache for holding multiple contexts and the high functional LUT (Look Up Table) that have the functions of multi-context and clustering. The context is a unit of configuration data for circuits. The rest of the paper is organized as follows: Section 2 presents the novel reconfigurable logic architecture we proposed. Section 3 details our experimental approach, and gives the evaluation models and the benchmark circuits. Section 4 discusses the effect of our logic architecture. Finally, we conclude in Section 5.

2. Proposal of reconfigurable logic architecture

2.1 Basic logic block architecture

Fig.1 depicts the architectural model of our logic block, which is a kind of the fine grained reconfigurable logic. Our logic block has configuration data cache (short for CDC) with LUT which realizes both the structure of a multi-context and multi-grain (short for MCMG-LUT) with clustering of LUTs.

This LUT has six constructive modes ((a)-(f)). The first three modes (a)-(c) are clustered LUTs. The latter three modes (d)-(f) are configured as multi-context and multi-grain LUTs. These modes are determined by each LUT. The CDC is the memory that holds more than one set of context data and mode bits for the MCMG-LUT, and even if LUT is working, the CDC can be rewritten by the data line only itself. The proposed logic block exploits these modes, hold promising the following effects; (1)Improving implementation density by the CDC, (2)Improving implementation efficiency by the clustered LUT, and (3) Reducing implementation area by the multi-context LUT.

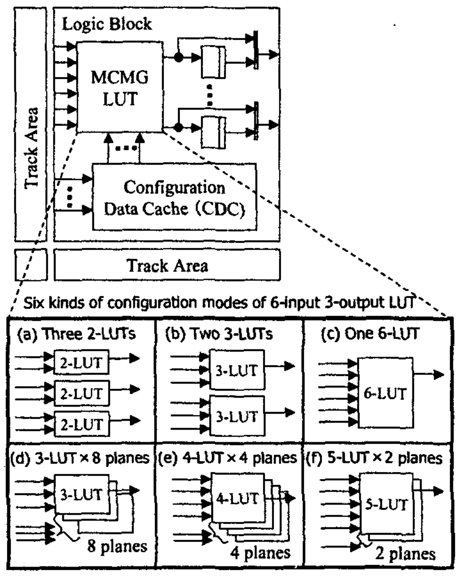


Figure 1. Logic block architecture for RC.

3. Experimental Methodology

In this section, we present a brief description of the area, delay and implementation efficiency modeling, and we define the implementation efficiency. Moreover, we describe benchmark circuits and evaluation flow. The area model[1] and delay model[2] was proposed by Rose et al. of University of Toronto and we modified part of this model for the architecture we propose.

3.1 Area Model

Area model is a simple model, consisting of the individual logic block with vertical and horizontal routing tracks that Fig.2 depicts it. It doesn't have a hierarchical routing structure. The number of the tracks of each routing channel (W) is given by,

$$W = a \times K + b, \quad (1)$$

where, K is the number of inputs to the LUT, a is the ratio coefficient of K and b is the number of fixed wires such as

*Presently, with Mitsubishi Electric Engineering Co., Ltd.

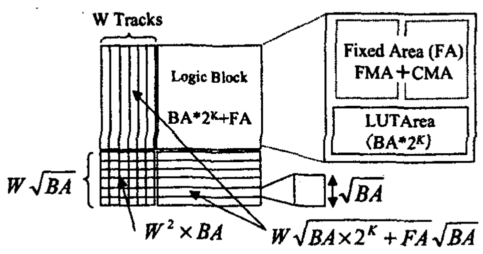


Figure 2. Area Model.

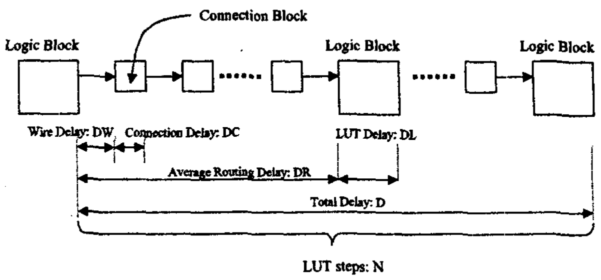


Figure 3. Delay Model.

clock signals, a reset signal and global wires. The width of each track is the square root of the bit area (*BA*), moreover, a logic block is assumed a square. A unit area in the evaluation model is the sum of the area of routing tracks that it join it with the logic block and the area of logic block. It can be expressed by following equations:

$$\begin{aligned} Area &= (W^2 \times BA + 2W\sqrt{BA \times 2^K + FA}\sqrt{BA}) + \\ &\quad (BA \times 2^K + FA), && (2)\\ FA &= FMA + CMA, && (3)\\ CMA &= BN \times BA + CLA. && (4) \end{aligned}$$

FA means the total area of the fixed domain in the logic block, and is broken down into *FMA* and *CMA*. *FMA* is the area of the fixed area such as *FF* , and *CMA* is the area for the CDC. Following, *CLA* is the area of control logic for MCMG-LUT, and *BN* is the number of bits for the CDC. A conventional FPGA is the same as when the area for the *CMA* equals zero.

We implement some benchmark circuits to our reconfigurable logic, and count the necessary number of logic blocks. Then we calculate a total area with the following equation:

$$Total\ Area = (\#\ of\ LBs) \times (Unit\ area\ of\ one\ LB). \qquad (5)$$

The area of a logic block is estimated using the Equation (2), and as we adopt $1.25\mu m$ CMOS process[1], we use the following parameters; *BA* is $400\mu m^2$, *FMA* is $5,100\mu m^2$, *CLA* is $2,250\mu m^2$, *a* is 1.0, and *b* is 12. We assume that *CLA* is $2,250\mu m^2$, because a breakdown of the total area of *CLA* includes three bits for mode changing ($1,200\mu m^2 = BA \times 3$) and the control logic ($1,050\mu m^2$). When we use these parameters, the area of a conventional logic block for 4-LUT($Area_4$) becomes $182,532\mu m^2$.

3.2 Delay Model

Delay model is shown in Fig.3, and we break down the routing delay (*DR*) into the sum of wiring delays (*DW*) and the sum of connection delays (*DC*). Then, *DL* is the logic block delay. *N* is the number of logic blocks between FFs on a critical path, and *D* is the total delay.

Searching for the number of logic blocks on the critical path of each benchmark circuit, the maximum delay can be approximated by the following equations:

$$\begin{aligned} DR &= \Sigma(DW) + \Sigma(DC), && (6)\\ D &= (DR + DL) \times N. && (7) \end{aligned}$$

In case of the routing structure not being clear, *DW* and *DC* are defined by,

$$\begin{aligned} \Sigma(DW) &= DR \times R_d, && (8)\\ \Sigma(DC) &= DR \times (1 - R_d), && (9) \end{aligned}$$

where, R_d denotes the delay coefficient when distributing the total routing delay to *DW* and *DC*.

We estimate the delay of the evaluation circuits by using Equation (10) and Equation (11). The ratio of the area of the logic block is given by

$$AR_{6/4} = \frac{\sqrt{LB\ Area\ of\ 6-LUT}}{\sqrt{LB\ Area\ of\ 4-LUT}}. \qquad (10)$$

Moreover the delay that includes routing delay for the CDC as a product of an area ratio and the routing delay by 4-LUT can be estimated by the following equation:

$$DR_6 = AR_{6/4} \times DR_4 \times R_d + DR_4 \times (1 - R_d). \qquad (11)$$

where, DR_4 denotes the routing delay of 4-LUT, and R_d is the wire-connection delay distribution coefficient. Each area uses the same values as mentioned above. As we adopte *$1.25\mu m$ CMOS process, we use the following parameters*[2]; DL_4 is $1.71ns$, DL_6 is $2.38ns$, DR_4 is $4.0ns$, and R_d is 0.1. We assume that the logic block delay of MCMG-LUT increased about 10 % more than the delay of a conventional 6-LUT as a rough estimate. The delay of MCMG-LUT(DL_{n6}) is $2.62ns$ accordingly. In this paper, we use this value for the evaluation of delay.

3.3 Implementation Efficiency Model

In implementing circuits into logic blocks, the amount of occupation bits (O-bits) and the amount of employment bits (E-bits) are defined by the following equations:

$$O - bits = (\#\ of\ logic\ blocks) \times 2^K, \qquad (12)$$

$$E - bits = \sum_{n=1}^{K}((\#\ of\ n - inputs\ logic\ blocks) \times 2^n). \qquad (13)$$

As a result, the implementation efficiency is obtained by

$$Implementation\ efficiency = \frac{(E - bits)}{(O - bits)} \times 100. \qquad (14)$$

3.4 Benchmark Circuits

The selection of the benchmark circuit is a very important problem. In this paper, we used the experimental circuits that are a selection of 10 logic synthesis benchmarks (alu4, ex1010, apex2, ex5p, des, frisc, dsip, seq, elliptic, and tseng) provided by Microelectronics Center of North Carolina (MCNC)[6] and six additional benchmark circuits. Table 1 shows our additional benchmark circuits. All additional circuits are designed with Verilog-HDL.

Table 1. Additional benchmark circuits.

Circuit	Description
ACS4	ACS Circuit of the Viterbi decoding
DIV8	Piplined Divider
FFT6	8 points FFT
DES	16 looped DES
MA8I	Multiply Accumulate operator
SCU	DLX processor(Instruction Decoder)

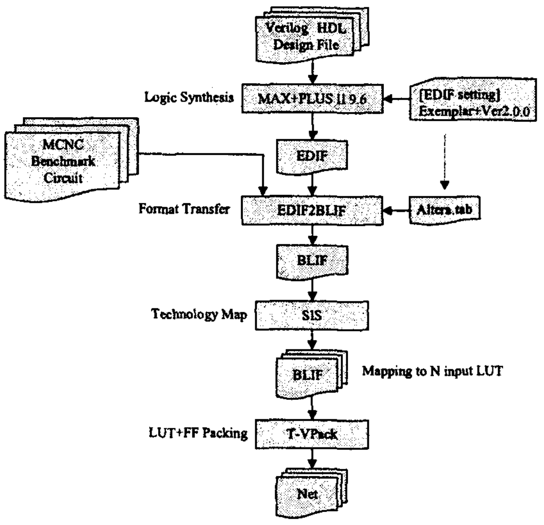


Figure 4. Architecture Evaluation Flow.

Table 2. Rate of various inputs actually used in each LUT.

Inputs	LUT Granularity				
	3-LUT	4-LUT	5-LUT	6-LUT	7-LUT
1 input	0.91	1.11	1.47	0.08	0.90
2 inputs	17.59	13.38	7.59	9.21	6.55
3 inputs	**81.50**	22.58	19.87	7.23	8.12
4 inputs	-	**62.94**	24.07	16.16	18.19
5 inputs	-	-	**47.01**	25.62	12.29
6 inputs	-	-	-	**41.70**	17.96
7 inputs	-	-	-	-	**35.98**

A unit is %.

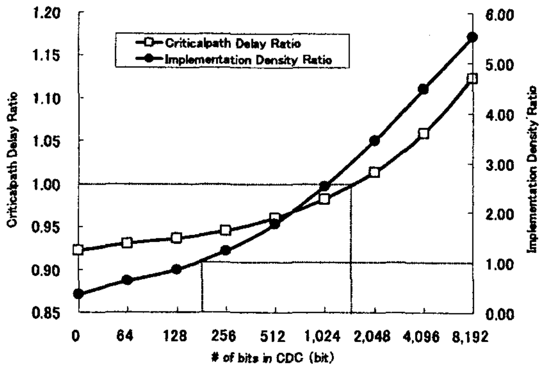


Figure 5. Implementation density and critical path delay.

3.5 Architecture Evaluation Flow

Implementation flow from source codes of Verilog-HDL to netlists is shown in the Fig.4. We got EDIF netlists by using MAX+PLUS II of the ALTERA from the benchmark circuit described with Verilog-HDL, and translated it into the BLIF format by using EDIF2BLIF [3] developed at the U.Toronto. Then, the BLIF netlists were mapped using SIS[4] of UCB from a 3-inputs-LUT to a 7-inputs-LUT, and it wase packed using TV-Pack[5] from U.Toronto, and finally got the netlists after the technology mapping.

4. Experimental results and discussions

4.1 Number of inputs on logic block

We examine the number of the input signals. Table 2 shows the rate of the input signals in benchmark circuits. The bold type is the rate of LUT mapped to the number of the maximum input of each LUT. The rate decreases as K increases. This means that there is surely an LUT for which the number of inputs is few even if the LUT granularity is enlarged, though this depends on the mapping algorithm.

4.2 Effect of the Configuration Data Cache

We implement the circuits to our logic block with mode (c) in order to evaluate the effect of the CDC.

The area of our logic block within CDC becomes larger than the area of 4-LUT. Accordingly, a wire delay becomes larger because the wire length between the logic blocks increases. A logic block delay becomes larger because our LUT is regarded as a 6-LUT too. However, the number of logic block stages on the critical path, which is the parameter N, decreases to change 4-LUT to 6-LUT. There is a trade off in these relations.

Fig.5 shows the implementation density and the critical path delay versus the number of bits in the CDC. They are the ratio of our logic block to a conventional 4-LUT logic block. The CDC does not increase only the implementation density, but also the critical path delay. However, from this graph, in the range between about 180 bits and about 1,600 bits in the CDC, our logic block is faster than a 4-LUT logic block and improves the implementation density. For instance, in the case of the CDC, 1,024 bits, the implementation density produces up to 2.52 times more than a 4-LUT logic block.

4.3 Effect of LUT Clustering

Our logic block can be clustered as three 2-LUTs (mode (a)) or two 3-LUTs (mode (b)). The LUT actual inputs are below the maximum inputs of LUT exist in some degree as shown in Table 2. For example, in the case of a 6-LUT, about 9.21% of the LUTs use only two inputs and 7.23% of the LUTs use only three inputs. It is very probable that LUT clustering reduces the number of LUTs actually being used.

Fig.6 shows the results from implemented benchmark circuits from 3-LUT to 7-LUT and our n6-LUT[1]. An implementation to n6-LUT is an ideal case. The normalized area shows the relative value against the minimum area. The normalized area of our n6-LUT approaches the minimum area. The implementation efficiency is improved 6% from a 6-LUT and became the level of a 5-LUT.

[1]MCMG-LUT which used the mode (a) and the mode (b) is called n6-LUT in order to distinguish from the conventional 6-LUT.

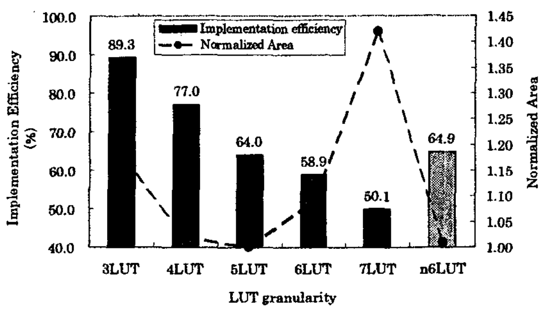
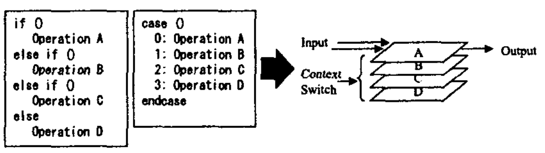


Figure 6. Normalized area and implementation efficiency.

4.4 Effect of Multi-context and Multi-grain LUT

In this paragraph, we consider the effect of multi-context and multi-grain mode. In the case of modes (d)-(f), there is the assumption that it doesn't activate at the same time in each context of LUT. Therefore, it can't be assured through a usual mapping method. In this evaluation, we adopt the mapping method shown in Fig.7. In this way, the independent block is implemented in another context, the independent block means the operation doesn't activate at the same time by Verilog-HDL syntax, such as the "case" sentence or the "if" sentence.



```
if 0
  Operation A
else if 0
  Operation B
else if 0
  Operation C
else
  Operation D
```

```
case 0
  0: Operation A
  1: Operation B
  2: Operation C
  3: Operation D
endcase
```

Figure 7. Mapping method for Multi-context LUT.

The ALU16 circuit of 8 operations and 16 bits precision, shows in the Fig.8, is used as the evaluation circuit. This circuit is implemented to a 7-LUT from a 3-LUT using conventional FPGAs, our 3-LUT with eight contexts (mode (d)) and our 4-LUT with four contexts (mode (e)). Each operation of ALU16 is implemented to our logic block by the flow of Fig.4, and we get a final netlist by being unified by the manual operation. The whole of ALU16 is implemented to the conventional LUT in accordance with Fig.4.

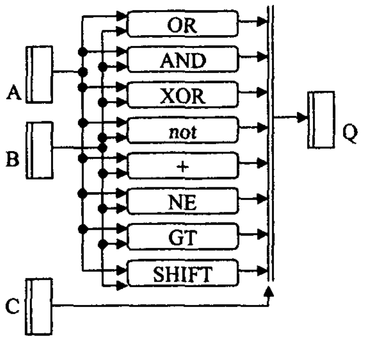


Figure 8. Structure of ALU16.

Fig.9 shows the relative area ratio of a benchmark circuit that we implement in five kinds of conventional LUTs and our two multi-context LUTs. Our multi-context LUT, in which the modes are eight 3-LUTs (mode (d)) and four 4-LUTs (mode (e)), doesn't include the CDC in this evaluation. As a result, the mode (d) shows the minimum area. It turns out that the area using mode (d) is about 87% of the implementation area using the conventional 4-LUT.

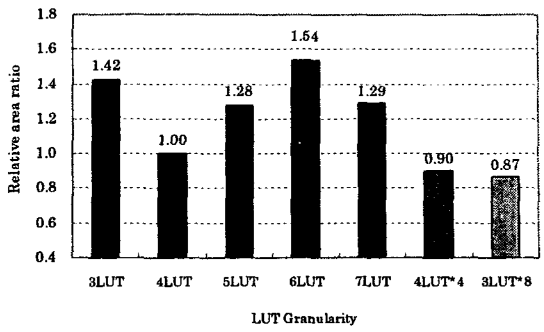


Figure 9. Area comparison of ALU16.

5. Conclusions and Future works

In this paper, we proposed a new architecture of logic block for the fine grained reconfigurable platforms that have configuration data cache for holding multiple contexts and the LUT that has the functions of multi-context and clustering. Compared to conventional logic blocks, our logic block improved the implementation density, the implementation efficiency and the implementation area, with the consequence that it brings more flexibility and high performance in the programmable device. We expect that the reconfigurable computing using our logic block architecture will become a key technology in the future multimedia systems or communication systems.

This research proposed and estimated the suitable logic block for RC. However, with the present CAE tool, it cannot implement in this logic block. Therefore, it is necessary to examine a CAE tool that can utilize this logic block effectively from now on. Moreover, details, such as how to replace configuration data and routing structure, are also due to be examined.

References

[1] J.Rose, R.Francis, D.Lewis : "Architecture of Field-Programmable Gate Arrays : The Effect of Logic Block Function on Area Efficiency," *IEEE J.Solid-State Circuits*, Vol.25, No.5, pp.1217-1225, 1992.

[2] J.Rose and S.Brown : "The Effect of Logic Block Architecture on FPGA Performance", *IEEE J.Solid-State Circuits*, Vol.27, No.3, pp.281-287, 1992.

[3] P. Leventis : "Using edif2blif Version 1.0 (Draft)," 1998. http://www.eecg.toronto.edu/~jayar/software/software.html

[4] E.M. Sentovich, et al. : "SIS:A System for Sequential Circuit Synthsis," Memorandom No. UCB / ERL M92/41, 1992. http://www-cad.eecs.berkeley.edu:80/Software/software.html

[5] A. Marquardt, et al. : "Using Cluster-Based Logic Blocks and Timing-Driven Packing to Improve FPGA Speed and Density," *ACM/SIGDA FPGA 99*, 1999. http://www.eecg.toronto.edu/~jayar/software/software.html

[6] S. Yang : "Logic Synthesis and Optimization Benchmarks, Version 3.0," *Tech. Report, Microelectronics Centre of North Carolina*, 1991.