

Logic synthesis and place-and-route environment for ORGAs

Minoru Watanabe and Fuminori Kobayashi
 Department of Systems Innovation and Informatics
 Kyushu Institute of Technology
 680-4 Kawazu, Iizuka, Fukuoka, 820-8502, Japan
 Email: { watanabe, fkoba }@ces.kyutech.ac.jp

Abstract—We have continued development of **Optically Reconfigurable Gate Arrays (ORGAs)** to realize larger virtual gate count VLSIs than currently available VLSIs. The grain and structure of ORGAs must be changed depending on their dynamically reconfigurable applications. Therefore, an ORGA development tool must be easily customizable for changes of the grain and structure. This paper presents an easily-customizable logic synthesis, and a place-and-route environment of ORGAs using appropriate software tools (Design Compiler and Apollo; Synopsys Inc.).

I. INTRODUCTION

Optically Reconfigurable Gate Arrays (ORGAs) with a holographic memory have been developed to realize virtual gate counts that are much larger than those of currently available VLSIs [1], [2]. To use ORGAs, development of tools is extremely important.

However, synthesis, and place-and-route tools for many programmable devices are specialized for specific target devices. Customizing them for other devices is usually difficult. Of course, FPGA tools, as well as SIS and VPR tools are available [3], [4], [5]. However, to adapt to a wide variety of fine grain architecture to coarse grain architecture, much effort is required.

Therefore, this paper presents an easily-customizable logic synthesis, place-and-route environment of ORGAs for any target using a logic synthesis tool and the place-and-route tool (Design Compiler and Apollo; Synopsys Inc.).

II. TOOL ENVIRONMENT FOR ORGAs

Apollo, Design Compiler, and Perl Script language are used to realize an easily-customizable logic synthesis, and place-and-route environment. The overall environment of our customized tool is shown in Fig. 1. In this environment, VHDL and Verilog languages are useful to produce reconfiguration contexts of an ORGA. Of course, C and C++ languages can also be used to develop the contexts by introducing a C or C++ compiler tool such as SpecC or SystemC in addition to them. The procedure progresses according to the following steps. First, a behavioral VHDL or Verilog design file is given to Design Compiler and the behavioral HDL design is converted to a structural HDL design in Design Compiler. Then, the structural HDL design is provided into Apollo and a place report file is output from Apollo. Finally, optical reconfiguration

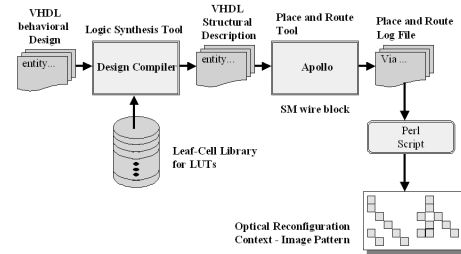


Fig. 1. Overall ORGA Tool environment.

TABLE I
 VARIETY OF LEAF CELLS IN LEAF-CELL LIBRARY.

Function NAME	Pin Input-Output	LUT type Input-Output
INV	X-Y	1-1-LUT
NAND2	X1,X2-Y	2-1-LUT
NOR2	X1,X2-Y	2-1-LUT
XOR2	X1,X2-Y	2-1-LUT
.	.	.
NAND3	X1,X2,X3-Y	3-1-LUT
.	.	.
NAND4	X1,X2,X3,X4-Y	4-1-LUT
WIOBIN	X	IOBI
WIOBOUT	Y	IOBO

format is generated by converting the place report file using Perl Script Language. Design Compiler functions to convert a behavioral HDL design to a structural HDL design based on the use of a leaf-cell library. Customizing the Design Compiler involves the production of a leaf-cell library that is suitable for a Look-Up Table (LUT) structure of ORGAs. A Look-Up Table (LUT) of target ORGA-VLSI chips can be programmed as a four-input one-output type. Therefore, often-used LUT patterns are defined as leaf cells that are stored in the leaf-cell library.

A place-and-route environment is constructed using Apollo. The customization of the tool is to define design rules and to design metal guards, and via guards.

Frame data corresponding to each leaf-cell in Design Compiler is constructed. The frame of a leaf-cell with one-input one-output is shown in Fig. 2. All vertical wiring is prohibited

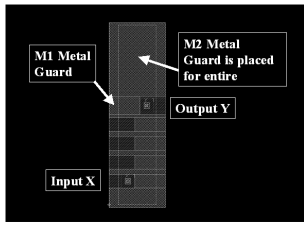


Fig. 2. Design of a Look-Up Table (LUT) frame.

by placing a vertical metal guard. Therefore, the LUT connection is realized using only horizontal wiring. In addition, because one horizontal channel is surrounded by horizontal metal guards, wiring across channels is not allowed. Although many leaf-cells are defined in Design Compiler, the necessary frame types of LUTs are only four: a one-input one-output LUT; a two-input one-output LUT; a three-input one-output LUT; and a four-input one-output LUT. Leaf-cells are placed into dedicated places of gate arrays. For the unplaced rest area, dummy leaf-cells are placed after leaf-cell placement.

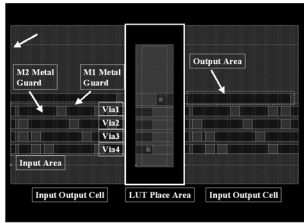


Fig. 3. Design of an Input-Output Cell frame including a LUT cell.

Input-Output Buffer Frame cells are placed outside of the gate array. The placement area is defined on dedicated places of the gate arrays. These cells are constructed using forbidden via areas and the wiring guard.

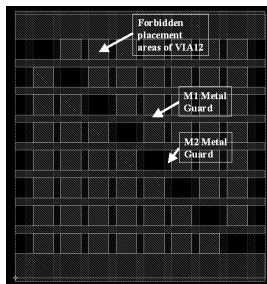


Fig. 4. Design of a switching-matrix frame.

The frame of a switching matrix is shown in Fig. 4. Each transmission gate corresponds to a via. The wiring channel of a switching matrix is guarded by a horizontal guard for the first metal layer and vertical guard for the second metal layer. Therefore, each wire is drawn along a wiring channel between the metal guards and does not jump across each wiring channel. The horizontal wire and vertical wire can be connected only through a via. Therefore, because via guards are implemented in seven places, except for one place, the

four-direction wiring and one-to-one connection of switching matrix structure can be realized.

III. SYNTHESIS AND PLACE-AND-ROUTE RESULTS

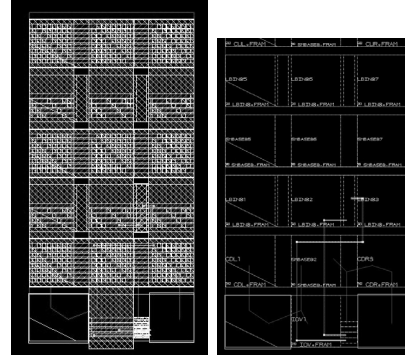


Fig. 5. Place-and-Route Result.

Using these tool environment, we have demonstrated a single inverter with a 2×2 logic block structure. The result is shown in Fig. 5. The correct routing is confirmed.

IV. CONCLUSION

This paper has presented easily-customizable logic synthesis and a place-and-route environment for ORGAs using a logic synthesis tool and a place-and-route tool (Design Compiler and Apollo, respectively; Synopsys Inc.). This is the first step of ORGA development environment. The implementation of an optimization between dynamically reconfiguring contexts remains as a task for future work.

V. ACKNOWLEDGMENT

This research was partially supported by the project of development of high-density optically and partially reconfigurable gate arrays under Japan Science and Technology Agency, the funds from the MEXT via Kitakyushu and Fukuoka innovative cluster projects, and the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Young Scientists (B), 16760275, 2005. The VLSI chip in this study was fabricated in the chip fabrication program of VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Rohm Co. Ltd. and Toppan Printing Co. Ltd.

REFERENCES

- [1] J. Mumbro, G. Panotopoulos, D. Psaltis, X. An, F. Mok, S. Ay, S. Barna, E. Fossum, "Optically Programmable Gate Array," SPIE of Optics in Computing 2000, Vol. 4089, pp. 763-771, 2000.
- [2] M. Watanabe, F. Kobayashi, "A 51,272-gate-count Dynamic Optically Reconfigurable Gate Array in a standard 0.35um CMOS Technology," International Conference on Solid State Devices and Materials, pp. 336-337, 2005.
- [3] E.M.Sentovich et al., "SIS: A system for sequential circuit synthesis," UC Berkeley ERL Memo. NO. UCB/ERL M92/4 (SIS_paper.ps of sis-1.2.tar.gz), 1991.
- [4] V.Betz, J. Rose, and A. Marquardt, "Architecture and CAD for Deep-submicron FPGAs, Kluwer Academic Publishers, 1999.
- [5] S. Kirkpatrick, C.D.Gelatt, M. P. Vecchi, "Optimization by Simulated Annealing," Science, No. 22, pp. 671-680, 1983.