Automatic Synthesis of Multilevel Combinational Logic

Andersen, Anders C.; Madsen, Jan; Madsen, J.R.; Pallisgaard, H

Published in:
Proceedings of the 15th European Solid-State Circuits Conference

Publication date:
1989

Document Version
Publisher's PDF, also known as Version of record

Automatic Synthesis of Multilevel Combinational Logic

A.C. Andersen, J. Madsen, J.R. Madsen, H. Pallisgaard

DesignCenter of Electronics Institute
Technical University of Denmark
DK2800 Lyngby, Denmark

Abstract

This paper describes a system for the synthesis of multilevel combinational logic, transforming functional description into mask layout. The system includes a logic synthesis part, partly consisting of tools developed at Eindhoven University of Technology, which has been interfaced to the layout synthesis part in the CATOE-system, developed at the DesignCenter of Electronics Institute. The various steps in the transformation are presented together with a complete design example, implementing a multi-output combinational decoder function.

1 Introduction

A traditional VLSI-implementation of multi-output combinational logic would be an array-structure e.g. a PLA. Module generators for such structural modules, based upon a abutment-based tile-layout style, are easy to implement. Moreover efficient optimization tools for PLA's have been available for a number of years. However such array-structures suffer from many electrical disadvantages due to long interconnections — i.e. the larger they get the slower they become.

Multilevel implementation in contrast offers many significant advantages in terms of flexibility — mapping options to different ASIC-technologies (eg. gate-array, standard cell and full custom), tradeoff-possibilities between area and speed, combined control- and datapath design etc. Targeting *full-custom* layout raises one serious problem — automatic layout- generation of multilevel logic is considerably more difficult than generation of PLA's.

The **CATOE**-system which has been developed at the DesignCenter of Electronics Institute, Lyngby, includes a basis for VLSI CAD-Tools **DOLPHIN**[1] utilizing process/technology encapsulation and simple design database management. This paper describes the organization of a system for automatic synthesis of multilevel combinational using the **CATOE**-system, and how the system has been used to design a chip realizing a multi-output combinational decoder function.

[1] Dynamic Object oriented Layout and Process abstract HIerarchical eNvironment

2 System Overview

The construction of the **DOLPHIN**-environment was initiated by the need of coordination of several different tool projects at the DesignCenter, covering logic- as well as layout synthesis domains. The environment should include user interfacing, process and design database -handling, thus it was decided to define a common design format to handle design data from highest (i.e. functional) to lowest (i.e. mask) -level. This format — **ALF** [MadJ89] — is ment as an internal design interchange format (ie. from one tool to another) in contrast to **EDIF**, which is intended for external interchange. **ALF** has a textual representation, to allow easy design interchange between the different computer systems on which the **CATOE**-system is implemented.

Figure 1 shows the transformation of a design from functional description down to mask level. The first step is handled by the logic synthesis tools, and it includes optimization and decomposition for multi-level logic synthesis[2]. Transformation from gate-level (logic level) to switch-level is handled by a netlist generator capable of producing a transistor netlist for each of the functions produced by the previous step. A cell-compiler is able to generate symbolic layout from the netlist description, and a compactor can produce the final mask layout. The implementation of the environment and some of the tools in it, are described in [AnMa89] and [AMMP88].

[2] The logic synthesis tools are developed at Eindhoven University of Technology, Nederlands [ThBe87].

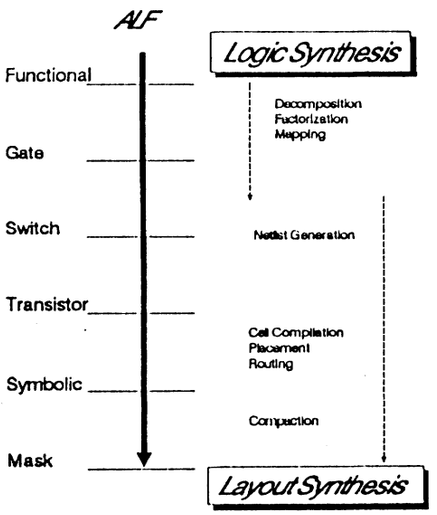

Figure 1: Transformation between different levels of abstraction in the **CATOE**-system. Design data are represented in **ALF**.

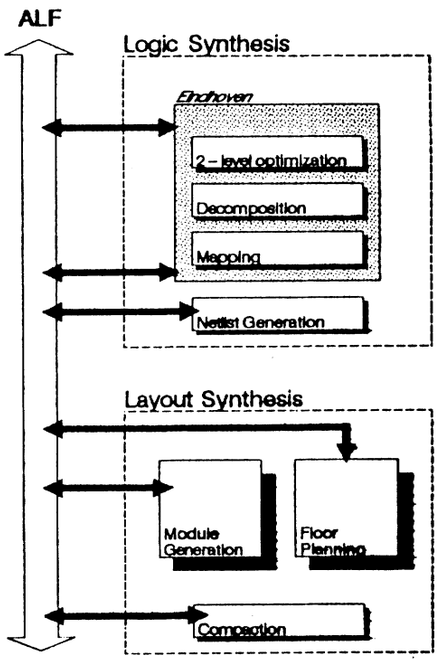

Figure 2: Encapsulation of the Eindhoven programs in the **CATOE**-system.

3 Logic Synthesis

The *logic synthesis* phase can be described as a transformation of an *abstract* description of logic (i.e. functional) to a gate-level description. Since the logic synthesis phase is to be followed by a layout synthesis phase using cell-compilation of the individual gates, we will extend logic synthesis to include netlist generation from the gate-level (logic level) description. The logic synthesis may now be divided into 3 steps:

- Multilevel Logic Optimization
- Mapping into target logic configuration
- Netlist generation for the individual gates

The optimization programs [ThBe87] offers optimization of boolean expressions based upon **ESPRESSO-II** and decomposition using the kernel factorization principles proposed by Brayton et al. [Bra82]. This program-package also includes a technology mapper [BeJe88] for fitting a set of optimized and decomposed functions to complex gates of a certain maximum *gate-size*. This package has been encapsulated in an **ALF**-interface [Brag89] and thereby included in the **CATOE**-system (see Figure 2). The logic synthesis tools will, from a functional **ALF** description as in Figure 3, synthesize a set of transistor netlist descriptions of the optimized and decomposed functions, as well as an overall circuit netlist.

Each (sub)function is realized as a complex gate. The netlist generation algorithm, which is not limited to pure combinational logic, is described in [Brag89].

4 Layout Synthesis

The *layout synthesis* transforms the netlists generated by the logic synthesis tools into mask layout. This part consists of 3 steps:

- Modulegeneration
- Floorplanning
- Compaction

Module Generation

The synthesis of complex gates is performed by the cell compiler **CELLO**, which takes an **ALF** transistor netlist description and transforms it into symbolic layout, also described in **ALF**.

CELLO produces a complementary static-CMOS layout in the line-of-diffusion layout style [UevC81]. In order to obtain area-optimized layout, **CELLO** uses an eulerpath evaluation algorithm, which is able to handle constraint pin placement, e.g. given from

```
(FUNCTION f
   (BOOLEAN Fout (NOT ( AND
         ( Ain ( OR Bin Cin ) (OR Din Ein Fin ))))
   (INTERFACE FOut Fout * OUT)
   (INTERFACE A Ain * IN)
   (INTERFACE B Bin * IN)
   (INTERFACE C Cin * IN)
   (INTERFACE D Din * IN)
   (INTERFACE E Ein * IN)
   (INTERFACE F Fin * IN)
   (INTERFACE Vdd Vdd * POWER Vdd)
   (INTERFACE Vss Vss * POWER Vss)
```

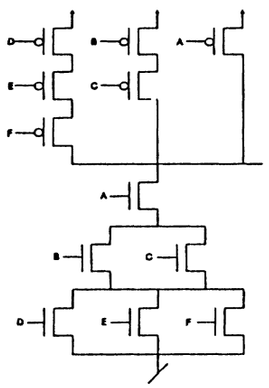

Figure 3: Logic synthesis in **DOLPHIN**; a) Functional description. b) Complex gate realization.

the floorplanner using the top-down design methodology. Further paramenters indicate whether the netlist should be regarded as fixed or changeable. The changeable netlist is used whenever no pretiming optimization have been performed, in order to enlarge the solution space. In this case **CELLO** tries to rearrange transistors in order to achieve a more compact solution. The fixed netlist is used in connection with a preprocessing timing optimization step making sure that the cell compiler does not change the timing. Figure 4 shows a **CELLO** generated layout of the function $f(a,b,c,d,e,f) = \overline{a(b+c)(d+e+f)}$ from Figure 3.

Floorplanning

The complex gates are placed in a standard cell like fashion, using rows of cells and channels. When all cells have been placed, they are interconnected using the two-layer channel router **CHAN**.

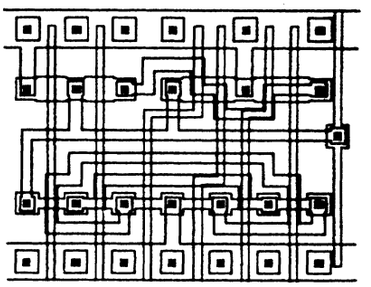

Figure 4: **CELLO** generated layout of the complex-gate from Figure 3

Compaction

After placement and routing the symbolic design is transformed into mask layout by a hierarchical virtual grid compactor **VIGI** [AMMP88]. **VIGI** first compacts each cell and then stretches individual cells to meet the abutment criterion within each row, i.e. connecting power supply. **VIGI** has been built on top of the object-oriented process abstraction **DRAGON** [AnMa89], which handles all data and information concerning the actual process. The technology information used inside **DRAGON** is loaded to the system through a complete design rule description. After compaction an **ALF** description of the final mask geometries exists and a CIF-dump of the total layout can be produced.

5 Design Example

The system described in this paper has been used to implement a decoder chip. The instruction-decoder from the industry standard microprogram sequencer **Am2910A** was chosen as an example for the first test chip generated by our system. The **Am2910A** has been a subject for educational purpose at the DesignCenter of Electronics Institute and further its instruction decoder is a good example of a multiple output combinational function suitable for demonstrating the different aspects of the tools within **DOLPHIN** and Eindhoven Optimization Package. Figure 5 shows a section of the chip core, while Figure 6 shows the final chip layout. The chip was produced in a 2μ nwell process[3].

[3] The chip was processed under the Nordic brokerage service NORCHIP.

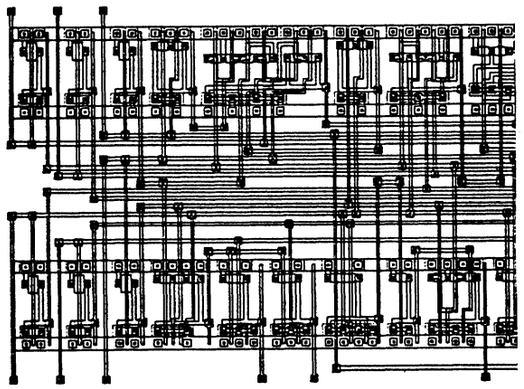

Figure 5: Section of the chip core.

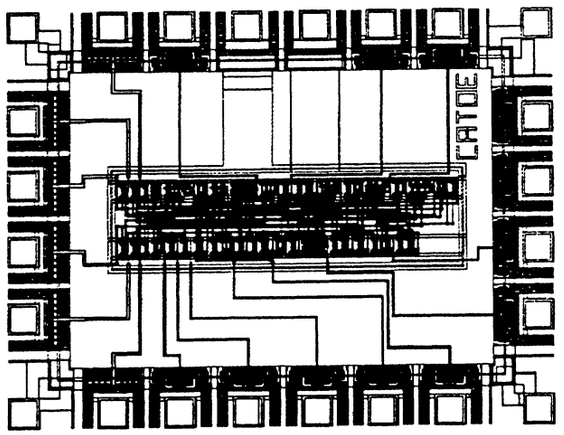

Figure 6: Chip layout of the **Am2910A** instruction decoder.

6 Summary

In this paper we have described a system for the synthesis of multilevel combinational logic. The system covers both logic synthesis and layout synthesis domains. The various steps in the transformation of multi-output combinational functions into mask layout have been described together with the design tools involved.

Further we have shown how a decoder chip has been completed using this system.

7 Acknowledgement

The authors would like to thank assoc. prof. O. Olesen for helpful suggestions and support of silicon area for realizing the test chip.

Parts of the tools mentioned in this paper are NORSILC-projects funded by Nordic Industrial fund and Teknologistyrelsen.

References

[AnMa89] Anders C. Andersen and Rene Madsen *"Subproject 10: Parameterized Cells," NORSILC report* March 1989.

[AMMP88] A.C. Andersen, J. Madsen, R. Madsen and H. Pallisgaard, *"A Dynamic Environment for VLSI Design Tools"*, NORSILC/NORCHIP seminar '88 , Copenhagen 26-27 oct. 1988.

[Brag89] Jens P. Brage *"Loss - A Logic Synthesis Sytem,"* Thesis report, DesignCenter of Electronics Institute, Technical University of Denmark, 1989.

[Bra82] R. K. Brayton and Curt McMullen *"The Decomposition and factorization of boolean expressions"*, ISCAS Proceding (April 1982) pp. 49-54.

[BeJe88] M.R.C.M Berkelaar and J.A.G. Jess *"Technology Mapping for Standard-Cell Generators," IEEE ICCAD-88.*, pp.470-473, 1988.

[MadJ89] Jan Madsen, *"ALF: Abstract Layout Format, the basis of an environment for VLSI software applications,"* DesignCenter of Electronics Institute, Technical University of Denmark, 1989.

[ThBe87] J.M.F Theeuwen and M.R.C.M. Berkelaar *"Logic Optimization with Technology and Delay in Mind"* Proceedings of the Internatinal Workshop on Logic Synthesis, May 1987.

[UevC81] T. Uehara and W.M. vanCleemput, *"Optimal Layout of CMOS Functional Arrays," IEEE Trans. Computer* Vol.c-30, pp.305-312, May 1981.