OPTIMIZATION OF NEGATIVE GATE NETWORKS REALIZED IN WEINBERGER-LIKE LAYOUT IN A BOOLEAN LEVEL SILICON COMPILER

Andrzej Wieclawski + , Marek Perkowski *

+ Warsaw Technical University, Institute of Electron Technology, Warsaw, Nowowiejska 15/19, Poland
* Portland State University, Department of Electrical Engineering, P.O. Box 751, Portland, Oregon, 97207, phone (503) 229-3806x27

1. INTRODUCTION

The random logic portion of a chip implementing a set of Boolean functions and sequential circuits usually represents a major contribution to chip area. Obviously, there are many circuits which realize the same Boolean function. Unfortunately, at present there is no general theory that provides designers (and design automation programs) with lower bounds for total area, for gate oxide area, and for delay time of logic implementations in integrated systems. Therefore, the main task for the computer optimization program appears in choice of the circuit with the most convenient layout.

DIADES is a design automation system with register-transfer level description on its input and CIF file on output [5]. The digital circuit can be described in both behavioral and structural mode. A set of successive compilations and hardware implementing and optimizing transformations create the description of the network on the level of logic gates and pass transistors. As the output of hardware compilation from the higher level, this description is usually nonoptimal and thus is next optimized by recursive technology independent transformations based on Boolean algebra (like A*0 = 0, A*A = A, etc.). Inverters are also inserted into long chains of AND or OR gates, being the results of iterative circuits' compilation [4].

The next stages are: technology dependant optimization of the logic network, and network's layout. It is assumed that the resultant network is multilevel, consists of complex negative gates, and is realized in semiregular Weinberger-style gate matrix layout. The logic minimization method intended for layout minimization is described in this paper.

It is assumed in our silicon compiler, that logic is constructed of n-channel, polysilicon gate MOSFET ratioless complex gates, performing any negative function. By negative function we understand negation of positive function, while positive function is any combination of AND and OR functors [1]. Functions for evaluation of circuit's performance parameters such as total area, area of gate oxide, and gate delay time are used. These functions are defined in terms of basic technology and selected topology parameters. The method can be adapted to other technologies.

2. OPTIMIZATION PROGRAM

2.1 Outline of the Program

The program is implemented in LISP 4.1 for CYBER 72 computer. The source network can include AND, NAND, OR, NOR, NOT gates and pass transistors. The initial phase of transformation applies structure improving transformations. Each operator transforms the selected segment of the logic network, that matches left pattern of transformation rule and fulfills certain conditions. The effectiveness of the transformation depends on the proper selection of operators. The optimization of the network is performed in a frame of search in solution trees, known from Artificial Intelligence.

The depth-first strategy with one successor is applied in the program. It means that for the last created node of the solution tree selected is any one applicable operator, the same is done for the successor node, etc. If there is no applicable operator to a solution tree's node - the network corresponding to this node is subject to the transformations of another type, that create complex gates. Next a backtrack in the solution tree is done and the procedure is iterated. The cost of the network with complex gates is calculated according to the cost functions derived in [3] and [4]. If this cost is smaller than the minimum cost of the solutions found until now, then the actual network becomes the solution. Else the solution remains unchanged.

To limit memory occupation only the actual network as well as its transformed parts are stored. To recreate the initial network, the inverse operators are applied in backtracking mode (technique known from AI languages like Micro-planner).

2.2 Transforming Operators

Typical graph of the network, that is scanned and processed by the operators, includes usually two logic levels. The goal of applying operators is:

- to maximize the amount of the tree-like segments with alternating connections of OR and AND gates,
- decreasing the number of gates,
- decreasing the number in inputs or outputs from gates (fan-in and fan-out).

The set of selected operators should permit for arbitrary transformation of the logic network. At the same time the number of the operators should be limited while else the optimization time increases too fast. We considered this trade-off by selecting operators for a set of practical networks.

There is currently 19 operators defined for networks without pass transistors, and additionally the same number for networks with pass transistors. The operators are based on logic laws like de Morgan's Theorem, and rules for shifting pass transistors around the network. The operators apply pattern matching in graphs, realized with an aid of Lisp data structures. The operators are generally

operator is so defined, that successive applications of the operator and its inverse operator do not change the network.

2.3. Optimization Algorithm

The optimization process follows the given algorithm:

1. Read the data base. Assign initial network cost ∞.
2. Find and store the applicable operators for the initial operator selected by the user in the initial node of the search tree.
3. If the selected operator cannot be applied (attempt of its application fail) then go to 20.
4. Store this operator and arguments for inverse operator in the new node of the tree.
5. Store this node and all applicable operators, that can follow the selected operator.
6. Actual node := new node.
7. Select first operator among those applicable in the actual node of the solution tree.
8. If there is no such operator, then go to 11.
9. Delete the selected operator from the set of applicable operators for the actual node.
10. If selected operator can be applied (attempt succeeded) then go to 4, else go to 7.
11. Store operators whose application leads to the actual node of the tree.
12. If the last executed operator was an inverse one then go to 18.
13. Apply an algorithm, transforming the actual network into a network of complex gates.
14. Calculate the cost of the solution with complex gates.
15. If this cost is not less than the minimum cost, then go to 18.
16. Store the last solution as the minimum solution.
17. Its cost is assigned as the minimum cost.
18. Apply the operator inverse to the last successfully applied forward operator.
19. If this is not the operator inverse to the initial operator, then go to 7.
20. Select the first operator among operators applicable in the initial node of the tree.
21. If there is such an operator, then go to 3, else end.

The effectiveness of the above algorithm depends on the selection of operators possible for application in the given node of the tree. The number of operators should be minimal, but we also want the quasiminimal solution not to differ much from the optimal one. Application of operators whose actions on the same element mutually anihilate should also be avoided. There are certain rules defined that describe possible sequences of operators. These rules can be interactively modified by the user for certain nodes depending on information about the circuit. They can also be automatically modified while searching the tree.

2.4 Transformation of Network to the Form Realized in MOS Technology

After application of operators described in 2.2, the network is optimized and ready for the second stage of transformations. At this stage the complex gates are created and all circuit redundancies removed. The terminal network cannot include the positive gates i.e., those that do not terminate with the inverter. The gates are transformed in the sequence of their types: OR, AND, NOR, NAND, NOT, INPUT, OUTPUT, FET, where: INPUT is the external input signal to the network, OUTPUT is the output signal from the network, and FET is the pass transistor. The above order results from the fact that only positive gates, AND and OR, can be used to create complex gates .Thus, if all positive gates have been used, there is no possibility of creating complex gates, which essentially simplifies further transformations of the network to the form realized in MOS technology.

The transformation of the given network's segment consists of checking the direct ancestors and successors of each gate, and transformation of this gate according to the information thus obtained. For instance, the OR gate can be either transformed to the complex gate (such as when gate leads its output only to AND or NAND gate) or to the negative gate (such as when joining to an inverter). The most of the possible variants of transformation exists for gates OR and AND, as well as for NOT and FET. For the second group, a large number of transformations result from the necessity of removing serial or parallel connections of inverters (or pass transistors of the same phase). Such situations are possible when, for instance, creating negative gates also creates additional NOT gates. It can also happen that because of too extensive restriction of the applicable operators in the given tree's node the discussed optimization algorithm has not removed the serial connection of NOT gates [3].

2.5. Cost of Complex Logic Gate

The cost functions require only the data known on the basis of the logic circuit and technological parameters assumed by the designer. The total cost of the complex logic gate is a weighted sum of total area cost, gate oxide area cost, and gate delay cost. The functions applied for evaluating the total area are based on typical gate layouts similar to [1], with design rules from [2].

3.CONCLUSION

The presented program can be applied for optimization of networks of hundreds of gates. Larger networks require higher processing time but the approach is amenable to parallel processors. The cost function and operators can be exchanged for optimization in new technologies and for conversion among technologies (for instance TTL to CMOS). More detailed description of the approach can be found in [3,4].

4.REFERENCES

1. I. Shirakawa, N.Okuda, T.Harada, S.Tani, H.Ozaki, "A Layout System for Random Logic Portion of MOS LSI".ACM 1980.
2. C. Mead, L.Conway, "Introduction to VLSI Systems", Addison-Wesley Publishing Company 1980.
3. A. Wieclawski, "Optimization of Logic Circuit in MOS Technology", Institute of Electron Technology, Warsaw Technical University, M.Sc.Th., 1982.
4. A. Wieclawski, M. Perkowski: "Optimization of Negative Gate Networks". Report. Portland State University, Department of EE, 1984.
5. M. Perkowski, E.B.Lee, Ch. Kim: "An Approach to Custom Design of VLSI Circuit Design Automation", Univ. of Minnesota, Department of EE, Report, 1982.