# Structural technology mapping with power optimization

Anna Fedotova
ISP RAS,
HSE
Moscow, Russia
aafedotova_6@edu.hse.ru

Mikhail Chupilko
ISP RAS,
Plekhanov RUE
Moscow, Russia
chupilko@ispras.ru

**Abstract—The paper is devoted to the implementation of structural technology mapper — a step of IC design flow based on standard cells — better (in terms of the results quality) than existing open-source logic synthesis tools. The main idea is to match input logic graph with standard cells represented as logic trees. A distinctive feature of our implementation from existing structural technology mapping approaches is the possibility to perform power-oriented optimization using a cost function based on specifications of physical characteristics of the target cells. A comparison with a technology mapper used in OpenLane has been performed on examples from OpenABC-D dataset.**

***Keywords— hardware design, integrated circuits, structural technology mapping, HDL, logic synthesis***

## I. INTRODUCTION

One of the main stages of VLSI design is logic synthesis [1] – construction of a logical circuit in terms of a given set of standard logical cells using an RTL model represented in a hardware description language (for example, in the Verilog language [2]). Logic synthesis process contains the following steps: translation of an RTL model into some internal representation, logic optimization, technology mapping into a netlist of standard cells taken from a technology library (described in the Liberty format [3]). In this paper, it is proposed to concentrate on the task of technology mapping of an optimized logic circuit into a netlist of standard cells.

In modern industrial practice, the problem of technology mapping is solved by commercial EDAs from Synopsys, Cadence Design Systems, and Siemens. In addition, it should be noticed that in existing open source solutions, such as OpenLane [4], the solving the problem of technology mapping is conducted by the Yosys [5] and ABC tools [6]. However, these tools are not able to compete with commercial CAD systems, since they do not support the design of complex VLSI circuits: they have restrictions on the size of the processed input circuits, as well as on considering the timing characteristics of the synthesized circuits. The typical size of circuits used in OpenLane does not exceed 10^6 gates.

Since all the open source solutions listed above use functional mapping, i.e. comparison of parts of an input design and standard cells using truth tables, it is interesting if it is possible to improve the characteristics of the technology mapping by using of structural comparison of standard cells and parts of the input logic circuit. It is assumed that the basic framework for developing a technology mapping tool is provided by ISP RAS, including tools for preparing logical circuits in internal representation obtained from files in Verilog format. Ultimately, a technology mapping tool should be developed that maps better (runs faster and synthesizes better estimates of netlist characteristics) of the existing open-source solution used in OpenLane.

The rest of the paper is organized as follows. The second chapter describes the related work. The third chapter is devoted to the description of the developed technology mapping implementation. The forth chapter shows the results of the experiments. The fifth chapter concludes the paper.

## II. RELATED WORK

The proposed in this paper structural mapper implementation uses approach similar to the one described in [7], which provides a solution to this problem in terms of pattern matching, description of technology-specific units, and optimization against technology-independent circuits represented as directed acyclic graphs.

In [7] it is suggested to use the Aho-Corasick algorithm to find the correspondence between input design and standard cells. Let a set of strings of total length `n` in the alphabet of size `k` be given. Aho-Corasick algorithm builds a prefix tree – trie for this set of strings, and then builds an automaton using this tree, which can be used in various string tasks – for example, to find all occurrences of each string from a given set in an arbitrary text in linear time [8]. Implementation described uses Aho-Corasick algorithm to match strings representing pattern paths with strings representing input circuit paths.

To the best of our knowledge, there is no more information on open-source structural technology mappers.

## III. SUGGESTED TECHNOLOGY MAPPER

Our implementation receives logical circuits transformed into the internal representation of the Utopia EDA project [10] of logic synthesis tool.

The Liberty file is read by the parser into a set of objects describing library cells, with access by object name or the entire collection. Then, for each cell, a truth table is built to organize the query “interface”: truth table -> cell name -> cell from a set of objects.

The internal representation uses gates and links to describe logic circuits. Gate types include: IN – input gates, OUT – output gates, ZERO – zero value gates, ONE – one value gates, BUF - buffer gates, AND – and gates, OR – or gates, XOR – xor gates, MAJ – major gates, NULL – null gates. The links between gates may or may not be inverted. The internal representation is used to describe different logical bases. An implementation of the representation is written in C++.

The algorithm in [7] (and follow exactly this limitation) can only match trees – the fanout of each logic gate does not exceed 1. Thus, it is necessary to transform the input circuit into a set of a number of trees. The algorithm finds all logic gates with fanout greater than 1 and makes “cuts” – unnecessary outgoing edges are removed and a new logic gate

is built in place of the deleted edge – an input and a new edge. The number of the gates for which the cut was made is saved for the correct assembly of the mapped trees into a single circuit after the completion of the algorithm.

The algorithm matches elements from the technology library with each of the trees in the set. Below, for clear understanding, it is described how the algorithm operates with only one tree (it works similarly for all the trees in the set).

The implementation uses a technology library of cells in Liberty format. Before the algorithm starts running, the cells from the library are translated into graphs in inner representation. Thus, all schemas that participate in the mapping are in inner representation.

Next, the Aho-Corasick algorithm is used. Aho-Corasick takes as input sets of strings representing circuits. Thus, to match a schema and a pattern-element of a technology library, it is necessary to match all the path lines of the pattern with a certain set of path lines of the input tree.

After all possible tree and library schema mappings have been found, information about the found mappings is saved. Dynamic programming is used to find the minimum tree coverage area (here can work different functions are based on information from Liberty library). Once the minimum coverage is found, information about the minimum coverage is saved. Information about the numbers of the logic gates of the input circuit, in the place of which each of the elements of the minimum coverage must be built is also stored. Thus, after the algorithm has passed through all the trees, information about all the minimum tree coverages is collected. Based on information about minimal coverages, a coverage of the entire input circuit is constructed.

## IV. Results of Experiments

We have conducted experiments on 15 designs taken from OpenABC-D repository [9]). The results (see Table 1) show that in the major number of cases (including one of the bigger aes design) we win in the value of power consumption estimated by OpenLane's OpenSTA. At the same time, we lose in area (not dramatically) and in time (the situation here is worse than with area).

## Conclusion

The goal of this research is to develop a technology mapper that would be more effective than existing open-source solutions in terms of power characteristics; at the same time it should have been based on structural mapping and allow to have a cost function based on Liberty technology library. In this paper we managed to show that it is possible to solve the task of technology mapping by means of structural approach and to receive a more efficient output designs according to the selected power cost function. The future work is to reduce its operational time, improve area and time values of the resulting mapping and to research a possibility of combination of the approach with truth-table-based mapping.

TABLE I. Results of Experiments

| *Design name* | *Vertex / edges number* | *Input/ Output number* | *Our technology mapper* | | | *Yosys+ABC* | | |
|---|---|---|---|---|---|---|---|---|
| | | | *Power, uW* | *Area, um²* | *The worst data arrival time, ns* | *Power, uW* | *Area, um²* | *The worst data arrival time, ns* |
| ac97_ctrl | 22 060 / 33 524 | 4 476 | 33 400 | 363 800 | 9.92 | **28 100** | **331 315** | **7.09** |
| aes | 39 215 / 68 140 | 1 212 | **6 660** | 188 356 | 121.79 | 14 900 | **136 239** | **6.94** |
| des3_area | 7 766 / 12 737 | 367 | **1 400** | 39 524 | 45.28 | 3 980 | **23 759** | **6.68** |
| fir | 9 002 / 13 560 | 761 | **1 000** | 38 655 | 33.8 | 8 710 | **33 143** | **7.51** |
| i2c | 2 018 / 3 187 | 305 | 629 | 9 238 | 15.68 | **434** | **5 577** | **4.58** |
| iir | 13 645 / 20 623 | 935 | **1 960** | 59 354 | 48.63 | 29 700 | **48 583** | **9.81** |
| mem_ctrl | 29 814 / 47 906 | 2 149 | 37 400 | 200 908 | 120.45 | 7 770 | **108 764** | **9.52** |
| pci | 38 279 / 57 826 | 6 586 | 75 100 | 670 748 | 270.7 | **59 200** | **620 715** | **12.24** |
| sasc | 1 214 / 1 827 | 260 | **354** | 5 291 | 8.82 | 406 | **3 700** | **3.32** |
| simple_spi | 1 764 / 2 694 | 296 | **478** | 7 754 | 11.87 | 484 | **4 805** | **3.29** |
| spi | 8 311 / 12 530 | 492 | 5 270 | 31 873 | **1.09** | **2 080** | **18 398** | 6.13 |
| ss_pcm | 762 / 1 165 | 194 | **278** | 3 692 | 8.64 | 281 | **2 367** | **2.19** |
| usb_phy | 893 / 1 380 | 222 | **219** | 4 415 | 5.47 | 239 | **2 729** | **2.43** |
| wb_conmax | 81 107 / 128 947 | 4 197 | 145 000 | 599 860 | 106.56 | **94 600** | **461 569** | **10.35** |
| wb_dma | 8 231 / 12 818 | 1 530 | 7 220 | 69 916 | 29.96 | **4 290** | **55 944** | **7.09** |

## References


[1] Kamkin A.S., Smolov S.A., Chupilko M.M. Comparison of open routes for designing digital equipment: qFlow, OpenLANE, Coriolis, SymbiFlow. Proceedings of the Institute of System Programming of the Russian Academy of Sciences. 2021; 33(6): pp. 111-130. DOI: 10.15514/ISPRAS-2021-33(6)-8

[2] IEEE Standard for Verilog Hardware Description Language. IEEE Std 1364-2005, 2006. DOI: 10.1109/IEEESTD.2006.99495

[3] Liberty standard "User Guides and Reference Manual Suite Version 2017.06"

[4] OpenLane. — URL: https://github.com/The-OpenROAD-Project/OpenLane

[5] Yosys. — URL: https://github.com/YosysHQ/yosys

[6] ABC. — URL: https://github.com/berkeley-abc/abc

[7] Keutzer K. DAGON: Technology Binding and Local Optimization by DAG Matching // 24th ACM/IEEE Design Automation Conference. 1987: pp. 341-347. DOI: 10.1145/37888.37940

[8] Aho-Corasick algorithm – URL: https://algorithmica.org/ru/aho-corasick (accessed: 05.02.2024).

[9] https://github.com/NYU-MLDA/OpenABC/tree/master/bench_openabcd

[10] Utopia EDA project – URL: https://gitlab.ispras.ru/mvg/utopia-eda.git