DOI: 10.15514/ISPRAS-2025-37(6)-52

Cut-based technology mapper with optimizations

[1,2] *M.M. Chupilko*, ORCID: 0000-0002-8772-5631 <*chupilko@ispras.ru*>
[1,2,3,4] *A.S. Kamkin*, ORCID: 0000-0001-6374-8575 <*kamkin@ispras.ru*>
[1,2] *D.R. Garyaev*, ORCID: 0009-0000-1059-4699 <*dgaryaev@gmail.com*>
[1,5] *E.S. Belin*, ORCID: 0009-0007-4687-3893 <*esbelin@ispras.ru*>
[5] *G.A. Mazov*, 0009-0004-9435-0454 <*gamazov@edu.hse.ru*>
[2,5] *V.S. Shtrenev*, ORCID: 0009-0003-4345-2154 <*vsshtrenev@ispras.ru*>

[1] *Plekhanov Russian University of Economics,*
36 Stremyanny lane, Moscow, 117997, Russia.

[2] *Ivannikov Institute for System Programming of the Russian Academy of Sciences,*
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.

[3] *Lomonosov Moscow State University,*
GSP-1, Leninskie Gory, Moscow, 119991, Russia.

[4] *Moscow Institute of Physics and Technology,*
9 Institutskiy lane, Dolgoprudny, Moscow Region, 141701, Russia.

[5] *Higher School of Economics Tikhonov Moscow Institute of Electronics and Mathematics,*
34, Tallinskaya st., Moscow, 123458, Russia.

Abstract. This paper addresses the problem of constructing an optimization-oriented technology mapper for logic synthesis. We present an implementation of a cut-based technology mapper developed within the Utopia EDA project, a prototype logic synthesis tool distributed under Apache 2.0 license. The proposed mapper is based on Boolean matching and supports multiple optimization objectives, including area (the total area of instantiated standard cells), power (the estimated total power consumption of the synthesized design), and timing (the estimated critical-path delay). It should be noted that targeting one objective implies accounting for constraints on the other two objectives. We provide a comparison with the technology mapper used in the OpenLane flow. Experimental results obtained on a benchmark set of thirty-one RTL designs (Verilog/SystemVerilog) demonstrate that, in the majority of cases, the proposed optimizations outperform the Yosys-based technology mapping used in OpenLane with respect to area and power. For timing optimization, the results are different, indicating directions for future work.

Keywords: technology mapping; integrated circuits; CAD design automation systems; logical synthesis; standard cells; optimization.

For citation: Chupilko M.M., Kamkin A.S., Garyaev D.R., Belin E.S., Mazov G.A., Shtrenev V.S. Cut-based technology mapper with optimizations. Trudy ISP RAN/Proc. ISP RAS, vol. 37, issue 6, part 4, 2025, pp. 85-96. DOI: 10.15514/ISPRAS-2025-37(6)-52.

Acknowledgements. This research was carried out at Plekhanov Russian University of Economics, and supported by the Russian Science Foundation, project No. 23-21-00313.

Подход к технологическому отображению с выбором направления оптимизации в процессе логического синтеза

[1,2] *М.М. Чупилко,* ORCID: 0000-0002-8772-5631 <*chupilko@ispras.ru*>
[1,2,3,4] *А.С. Камкин,* ORCID: 0000-0001-6374-8575 <*kamkin@ispras.ru*>
[1,2] *Д.Р. Гаряев,* ORCID: 0009-0000-1059-4699 <*dgaryaev@gmail.com*>
[1,5] *Е.С. Белин, ORCID:* 0009-0007-4687-3893 <*esbelin@ispras.ru*>
[5] *Г.А. Мазов,* ORCID: 0009-0004-9435-0454 <*gamazov@edu.hse.ru*>
[2,5] *В.С. Штренев,* ORCID: 0009-0003-4345-2154 <*vsshtrenev@ispras.ru*>

[1] *Российский Экономический Университет им. Г.В. Плеханова,*
Россия, 115054, г. Москва, Стремянный переулок, д. 36.

[2] *Институт системного программирования им. В.П. Иванникова РАН,*
Россия, 109004, г. Москва, ул. А. Солженицына, д. 25.

[3] *Московский государственный университет имени М.В. Ломоносова,*
119991, Россия, Москва, Ленинские горы, д. 1.

[4] *Московский физико-технический институт,*
141701, Московская область, г. Долгопрудный, Институтский переулок, д. 9.

[5] *Московский институт электроники и математики (МИЭМ ВШЭ),*
Россия, 123458, г. Москва, Таллинская ул., д. 34.

Аннотация. В данной статье мы фокусируемся на проблеме технологического отображения, как этапа логического синтеза цифровых СБИС. В основе подхода лежит сопоставление частей исходной логической схемы и элементов технологической библиотеки по таблицам истинности (т.н. Boolean matching). Особенностью предлагаемого подхода является возможность выбора стратегии оптимизации (площадь – как сумма площадей экземпляров технологических ячеек, задержка – как длина критического пути в схеме, потребляемая отображенной схемой мощность – как сумма статического и динамического потребления выбранных технологических ячеек) при удовлетворении ограничений на два других параметра. Подход реализован в инструменте Utopia EDA, распространяемом по лицензии Apache 2.0. В работе показаны результаты проведенных экспериментов над тридцатью одной RTL-моделью, реализованных на языках Verilog/SystemVerilog, с использованием САПР OpenLane. Эксперименты показали, что реализация достижения целевой стратегии в рамках нашего подхода позволяет в большинстве случаев в стратегиях «площадь» и «энергопотребление» достичь результаты лучшие по целевой характеристике, чем достигает ПО Yosys, входящее в САПР OpenLane в качестве средства логического синтеза и технологического отображения. Случай стратегии «задержка» показывает направление дальнейших работ.

Ключевые слова: технологическое отображение; сверхбольшие интегральные схемы СБИС; системы автоматизации проектирования САПР; логический синтез; логическая схема; оптимизация.

Для цитирования: Чупилко М.М., Камкин А.С., Гаряев Д.Р., Белин Е.С., Мазов Г.А., Штренев В.С. Подход к технологическому отображению с выбором направления оптимизации в процессе логического синтеза. Труды ИСП РАН, том 37, вып. 6, часть 4, 2025 г., стр. 85–96 (на английском языке). DOI: 10.15514/ISPRAS–2025–37(6)–52.

Благодарности: Исследование выполнено в РЭУ им. Г.В. Плеханова при поддержке Российского научного фонда, проект № 23-21-00313.

1. Introduction

The process of integrated circuit (IC) fabrication begins with the development of detailed specifications that define the required characteristics and functionality of the target IC. Based on these specifications, a design is created using a hardware description language such as Verilog or VHDL; in this work, we assume the use of Verilog. The design produced at this stage is referred to as an HDL model. The HDL model must then be verified against its specifications during the

functional verification stage. Once verification is completed successfully, the process proceeds to logic synthesis.

Logic synthesis produces a technology-independent representation that implements the specified functionality. The first step of this process involves parsing the Verilog description and constructing an internal graph-based representation. A widely used representation is the And-Inverter Graph (AIG), which is a compact model of Boolean functions composed of two basic elements: AND gates and inverters. In an AIG, nodes represent AND gates, while edges may be either direct or inverted, the latter indicating logical negation. This representation is particularly well suited for efficient logic optimization and formal verification.

After logic optimization, the next stage is technology mapping, which constitutes the main focus of this paper. In our study, we assume a target technology based on standard cells described in the Liberty format. We focus on functional technology mapping, in which the optimized AIG representing the desired functionality is transformed into a new graph by matching the truth tables of standard cells to subgraphs of the AIG, commonly referred to as *cuts*. Our analysis indicates that existing technology mappers have insufficient support for selection of optimization objectives. To address this limitation, we propose a new method for constructing optimization-oriented technology mappers and describe the development of its implementation.

The remainder of the paper is organized as follows. Section 2 reviews related work. Section 3 presents details of the proposed implementation. Section 4 discusses experimental results and provides a comparison with the standard technology mapper used in OpenLane. Finally, Section 5 concludes the paper.

2. Related work

Technology mapping in digital circuit design has been shaped by numerous contributions, among which the work of Alan Mishchenko [1] is particularly significant. The core idea of his approach lies in the preliminary selection of an optimal cut–defined as a subset of nodes that isolates a portion of the circuit and captures a sub-function for targeted optimization–without relying on the physical parameters of cells from the technology library.

The algorithm begins by computing *K-feasible cuts* for each vertex of an AIG graph, where K-feasible denotes a partition of the graph containing no more than K leaves. After identifying these cuts, the vertices are processed in topological order, from the primary inputs (PIs) to the primary outputs (POs). At this stage, the algorithm evaluates the delay associated with each cut and selects the cut that minimizes depth, which is essential for timing optimization. Following timing optimization, the algorithm performs area recovery to reduce the overall gate count and thus optimize circuit area. This step employs two complementary strategies: *area flow* and *exact local area*. Area flow promotes logic sharing across different functions, while exact local area minimizes the number of gates within individual cuts. Once area recovery is completed, the algorithm determines the best cover for the AIG by traversing the graph in reverse topological order, from the primary outputs back to the primary inputs. This ensures that earlier optimization decisions are preserved and not adversely affected by subsequent steps. The final result of this process is a mapped netlist that specifies the optimized circuit in terms of concrete logic gates and their interconnections, making it suitable for further simulation or fabrication.

This methodology is implemented in the open-source ABC logic optimization tool [2], which operates in conjunction with the open-source Yosys logic synthesis framework [3]. At present, Yosys serves as the sole logic synthesis engine employed by the open-source IC design flow OpenLane [4].

Several other open-source tools are also available. For example, Mockturtle [5] is a C++ library for logic synthesis and optimization that supports multiple network representations, including AIGs and Majority-Inverter Graphs (MIGs), and provides a variety of optimization techniques. In Mockturtle, technology mapping is performed in a manner similar to Mishchenko's algorithm, but with notable

extensions. In particular, Mockturtle explores multi-output cuts [6] and combines Boolean and *structural matching* techniques [7].

Despite their effectiveness, existing algorithms exhibit several limitations that motivate the present work. For instance, it is generally not possible to explicitly optimize for delay, area, or power with guaranteed outcomes. Instead, users are restricted to selecting predefined optimization heuristics (such as area flow), whose impact on the target metrics cannot be guaranteed. The goal of our ongoing research is to address this limitation by developing a technology mapping framework that provides optimization with guarantees across multiple objectives.

3. Description of the approach

The proposed technology mapping approach is presented below. It consists of a forward pass, in which the solution space is constructed (see Fig. 1), and a backward pass, in which the best solution is selected with respect to the target objective (see Fig. 2).

- **solution = {}** – an empty mapping from cells to pairs *(cost, match)*.
- for each **cell** of the circuit (in topological order from inputs to outputs):
  - if **cell** is a primary input:
    - solution[cell].cost = 0;
    - solution[cell].match = (IN, empty);
  - otherwise:
    - solution[cell].cost = ∞;
    - solution[cell].match = null;
    - for each **subnet** (from the given class) whose output is *cell* and for which all inputs already have matches, i.e., solution[input].match ≠ null:
      - for each **technology cell** *celltype* corresponding to this subnet:
        - cost = estimate(celltype, subnet, solution);
        - if cost < solution[cell].cost:
          - solution[cell].cost = cost;
          - solution[cell].match = (celltype, subnet).

Fig. 1. Forward pass (solution space construction).

- **queue = {}** – an empty queue of circuit cells.
- for each **primary output** of the circuit:
  - queue.enq(output);
- while queue ≠ {}:
  - cell = queue.deq();
  - if solution[cell].match = null:
    - error – technology mapping was not found;
    - invoke recovery mechanisms (see separate document);
  - otherwise:
    - mark the cell *cell*;
    - for each input *input* of the subnet in solution[cell].match.subnet:
      - if *input* has not been visited:
        - queue.enq(input);
- for each **marked cell** (in topological order):
  - create a technology cell of type solution[cell].match.celltype, connecting it to the cells created for the inputs of solution[cell].match.subnet.

Fig. 2. Backward Pass (netlist reconstruction from the solution space).

The technology mapping implementation was developed as part of the Utopia EDA project [8], which is distributed under the Apache 2.0 license. The design flow accepts Verilog/SystemVerilog files as input. Verilog designs are processed using Yosys as a front end, which generates an RTL Intermediate Language (RTL IL) representation. This RTL IL is subsequently converted into Utopia EDA's internal representation (IR) as an object of the *Design* class.

The Design object is then decomposed into instances of the *Subnet* class. Each Subnet contains only combinational logic, while flip-flops and latches are stored separately. The resulting set of subnets undergoes logic optimization and technology mapping to the target process design kit (PDK). Sequential elements are mapped to PDK independently from subnets. After mapping, the combinational subnets and sequential elements are recombined to produce the final netlist.

The technology mapping procedure is driven by an optimization objective, which can take one of three possible values: area, delay, or power. Each objective is associated with a corresponding cost function that is used to evaluate candidate matches during technology mapping.

The area cost function is defined as:

$$\mathrm{AreaFlow}[c] = \frac{\mathrm{AreaFlow}[\mathrm{input}_1] + \cdots + \mathrm{AreaFlow}[\mathrm{input}_m] + \mathrm{area}(c)}{\mathrm{fanout}(c)},$$

where $\mathrm{AreaFlow}[c]$ denotes the *area flow* propagated through each fanout of cell c, input_i is the i-th input of cell c, $\mathrm{area}(c)$ is the area of cell c, and $\mathrm{fanout}(c)$ is the number of fanouts of cell c.

The power cost function follows the same flow-based principle:

$$\mathrm{PowerFlow}[c] = \frac{\mathrm{PowerFlow}[\mathrm{input}_1] + \cdots + \mathrm{PowerFlow}[\mathrm{input}_m] + \mathrm{power}(c)}{\mathrm{fanout}(c)},$$

where $\mathrm{PowerFlow}[c]$ represents the *power flow* propagated through each fanout of cell c, and $\mathrm{power}(c)$is the power consumption of cell c.

The delay cost function is defined in a straightforward manner:

$$\mathrm{ArrivalTime}[c] = \max\{\mathrm{ArrivalTime}[\mathrm{input}_1], \ldots, \mathrm{ArrivalTime}[\mathrm{input}_m]\} + \mathrm{delay}(c),$$

where $\mathrm{ArrivalTime}[c]$ denotes the signal arrival time at the output of cell c, and $\mathrm{delay}(c)$ is the intrinsic delay of the cell.

4. Experiments results

The objective of our experiments is to evaluate the differences between the results produced by our optimization-oriented technology mappers and those obtained using the baseline technology mapper in OpenLane, which is based on Yosys coupled with ABC and employs various optimization passes.

We conducted a series of experiments using 31 open-source RTL designs as input circuits for technology mapping. For each design, we estimated the worst-case arrival time (hereafter referred to as *delay*), area, and power based exclusively on the results of logic synthesis, without performing physical synthesis or place-and-route steps.

To generate the baseline netlists for each design, we used Utopia EDA as follows:

```
read_verilog --top <top-module-name> <file1> … <fileN>
logopt dcrx fr cp aig resyn2
write_verilog netlist.v
write_verilog verilog_lib.v
```

The script above reads the design files, applies basic logic optimizations–such as dead-component removal, functional reduction, constant propagation, AIG transformation, and a sequence of other predefined optimizations–and produces two output files:

1. netlist.v – a Verilog file containing the elaborated top-level module;

2. verilog_lib.v – a Verilog file containing definitions of internal types.

Since Utopia EDA uses Yosys as its Verilog front end, the read_verilog command internally invokes the following Yosys passes: proc, opt, memory, flatten, and opt -fast.

To generate the technology-mapped netlist using Utopia EDA, the following script was applied:

```
read_verilog --top <top-module-name> netlist.v verilog_lib.v;
read_liberty <lib-file>
techmap --min-pass-num 8 --max-cut-num 6 --max-fanout 8 \
--objective <area|delay|power> --delay-constraint <clock-period>;

[insbuf; resize;]

stat_design
write_verilog utopia.v;
```

To perform technology mapping with Yosys (ABC) the following additional passes were applied:

```
techmap
dfflibmap -liberty <lib-file>
abc -D <clock-period> -constr <sdc-file> -liberty <lib-file> -script <ABC-script>
stat -t <top-module-name> -liberty <lib-file>
write_verilog yosys.v
```

For each optimization objective, a separate ABC script was used. The following definitions are common to all three technology-mapping scripts:

```
set abc_elab "read_constr,<sdc-file>;strash;scleanup;"
set abc_fine_tune "buffer,-N,8;upsize,{D};dnsize,{D}"
set abc_map_old_dly "map,-p,-B,0.2,-A,0.9,-M,0"
set abc_map_new_area "amap,-m,-Q,0.1,-F,20,-A,20,-C,5000"
set abc_resyn2 "balance; drw,-l; drf,-l; balance; drw,-l; drw,-l,-z; balance;
drf,-l,-z; drw,-l,-z; balance"
set abc_choice2 "fraig_store; balance; fraig_store; ${abc_resyn2}; fraig_store;
${abc_resyn2}; fraig_store; ${abc_resyn2}; fraig_store; fraig_restore"
```

The final ABC command sequences used for area, delay, and power optimization are listed below:

```
1) set area_map "+${abc_elab};${abc_map_new_area};&get,-
   n;&st;&dch;&nf;&put;${abc_fine_tune}";
2) set delay_map "+${abc_elab};${abc_map_old_dly};&get,-
   n;&st;&dch;&nf;&put;${abc_fine_tune}";
3) set power_map
   "+${abc_elab};${abc_map_new_area};${abc_choice2};${abc_map_new_area};&ge
   t,-n;&st;&dch;&nf;&put;${abc_fine_tune}";
```

To evaluate the characteristics of the mapped designs, we substituted the synthesized netlists in the OpenLane flow with our mapping results and then executed the subsequent steps that generate timing and power reports. The area of the mapped designs was extracted directly from the synthesis tools, as all flows produce consistent area values at this stage.

Detailed results are presented in Table 1. Each design was synthesized three times using (i) OpenLane with Yosys, (ii) Utopia EDA, and (iii) Utopia EDA with buffering. For each flow, separate runs were performed with area-, delay-, and power-oriented optimization settings of the corresponding logic synthesis tool (denoted in the table as the optimization objective: area, delay, or power). The best absolute value of each characteristic is highlighted in bold. If the value obtained with *OpenLane + Utopia EDA + buffering* is better than that obtained with *OpenLane + Yosys*, it is additionally underlined.

Table 1. Comparison of synthesis results between Yosys and Utopia EDA using OpenLane

Design number/ Design name	clk per.	Obj.	Yosys (i)			Utopia EDA (ii)			Utopia+buffering (iii)			Ratio (1)/(2)		
			area, um2	delay, ns	power, mW	area, um2	delay, ns	power, mW	area, um2	delay, ns	power, mW	area	delay	po-wer
1) OpenSpike [9]	19	area	2 721 369	68,61	348,00	**1 635 302**	248,99	206,00	2 478 894	5,27	96,60	1,10	13,02	3,60
	19	delay	2 964 339	68,78	543,00	1 955 998	248,99	188,00	2 479 923	**5,05**	94,50	1,20	13,62	5,75
	19	power	2 717 820	68,46	371,00	1 802 526	248,99	174,00	2 762 806	5,13	**94,40**	0,98	13,35	3,93
2) TT06 [10]	11	area	43 699	7,62	10,80	**26 562**	19,00	8,50	37 201	**6,67**	**5,94**	1,17	1,14	1,82
	11	delay	49 688	6,97	15,00	37 285	12,03	8,43	39 762	7,18	6,97	1,25	0,97	2,15
	11	power	42 716	7,77	10,20	30 133	7,61	7,46	41 784	7,05	7,31	1,02	1,10	1,4
3) ac97 ctrl [11]	6	area	169 865	4,17	21,00	**91 756**	5,62	**14,10**	102 717	2,31	14,30	1,65	1,81	1,47
	6	delay	179 357	4,17	21,00	102 576	5,94	**14,20**	106 465	**2,22**	14,30	1,68	1,88	1,47
	6	power	169 704	4,17	20,90	95 555	8,67	**14,10**	103 547	2,45	**14,20**	1,64	1,70	1,47
4) aes core [12]	13	area	165 205	4,65	48,00	**80 141**	5,99	**30,30**	151 424	4,37	38,30	1,09	1,06	1,25
	13	delay	208 111	**3,88**	70,70	126 037	8,19	52,00	151 366	4,33	38,30	1,37	0,90	1,85
	13	power	164 251	4,49	48,70	91 415	7,31	35,50	166 277	5,01	40,70	0,99	0,90	1,20
5) ddr3 controller [13]	18	area	241 549	5,99	18,90	**132 428**	29,08	10,40	176 055	7,05	**10,30**	1,37	0,85	1,83
	18	delay	247 785	**5,81**	20,20	191 566	70,38	14,30	192 259	6,55	11,00	1,29	0,89	1,84
	18	power	242 623	5,88	19,50	137 705	50,64	**10,10**	162 921	7,16	**10,20**	1,49	0,82	1,91
6) des area op [14]	15	area	44 587	5,46	4,72	**18 004**	10,02	2,83	24 696	5,11	**1,54**	1,81	1,07	3,06
	15	delay	55 519	**4,65**	5,99	30 030	9,34	4,46	28 509	5,38	1,82	1,95	0,86	3,29
	15	power	43 805	5,50	4,54	20 201	10,99	3,27	28 848	5,37	**1,62**	1,52	1,02	2,8
7) des perf opt [15]	12	area	896 435	3,89	175,00	**443 706**	43,71	106,00	489 601	**3,53**	92,60	1,83	1,10	1,89
	12	delay	1 107 511	**3,56**	225,00	650 364	48,50	167,00	502 153	**3,62**	94,80	2,21	0,98	2,37
	12	power	888 256	3,95	173,00	478 959	44,35	120,00	542 962	**3,58**	**91,20**	1,64	1,10	1,90
8) ethernet [16]	19	area	854 025	10,72	36,40	**446 752**	145,94	32,60	469 080	**4,96**	**24,50**	1,82	2,16	1,49
	19	delay	878 290	9,68	36,70	558 391	145,64	33,80	611 734	**5,09**	**24,70**	1,44	1,90	1,49
	19	power	869 264	10,58	37,30	461 046	147,21	25,40	504 601	**5,01**	**24,80**	1,72	2,11	1,50
9) i2c [17]	7	area	13 022	3,85	1,46	**7 340**	3,22	**0,93**	9 707	2,85	1,45	1,34	1,35	1,01
	7	delay	14 236	2,95	1,47	9 687	2,75	1,08	10 807	**2,48**	1,89	1,32	1,19	0,78
	7	power	12 711	4,88	1,44	7 891	5,48	**0,932**	10 796	3,02	1,57	1,18	1,62	0,92
10) mem ctrl [18]	17	area	104 500	8,28	12,40	**57 320**	15,04	**2,61**	81 885	7,31	**2,66**	1,28	1,13	4,66
	17	delay	115 282	**5,52**	13,90	74 727	11,87	2,72	84 152	6,84	2,72	1,37	0,81	5,11
	17	power	103 955	8,40	9,12	67 102	15,44	2,70	91 060	7,82	**2,66**	1,14	1,07	3,43
11) oscc plic [19]	19	area	54 794	7,56	4,87	**29 944**	9,3	3,65	33 087	7,85	**2,50**	1,66	0,96	1,95
	19	delay	57 596	7,56	4,41	38 456	**7,22**	3,50	34 733	**7,27**	2,58	1,66	1,04	1,71
	19	power	52 897	7,56	4,46	32 972	8,27	3,81	37 380	7,68	**2,46**	1,42	0,98	1,81
12) oscc psram [20]	7	area	16 426	3,49	4,41	**8 230**	8,4	2,84	**11 027**	3,72	2,11	1,49	0,94	2,09
	7	delay	17 277	**2,87**	4,72	11 517	16,66	5,46	12 214	4,00	1,84	1,41	0,72	2,57
	7	power	16 123	4,13	4,63	9 215	8,87	3,32	12 043	4,11	**1,48**	1,34	1,00	3,13
13) oscc vga [21]	4	area	21 663	11,00	9,70	**13 858**	25,70	**2,25**	18 912	23,97	4,05	1,15	0,46	2,40
	4	delay	24 722	**7,80**	13,10	18 939	20,52	2,71	20 611	22,95	4,37	1,2	0,34	3,00
	4	power	20 042	11,36	6,87	15 496	28,96	2,62	21 502	26,39	4,14	0,93	0,43	1,66
14) pci [22]	15	area	266 416	26,67	9,06	**139 991**	28,78	5,39	150 989	4,40	5,50	1,76	6,06	1,65

Design number/ Design name	clk per.	Obj.	Yosys (i)			Utopia EDA (ii)			Utopia+buffering (iii)			Ratio (1)/(2)		
			area, um2	delay, ns	power, mW	area, um2	delay, ns	power, mW	area, um2	delay, ns	power, mW	area	delay	po-wer
	15	delay	280 504	26,67	9,15	191 969	24,49	**5,40**	187 823	**4,24**	5,51	1,49	6,29	1,66
	15	power	268 156	26,67	9,07	144 889	29,99	**5,39**	165 421	4,86	5,49	1,62	5,49	1,65
15) ravenoc [23]	23	area	662 995	21,61	15,70	**540 766**	40,24	10,80	614 891	15,57	**9,34**	1,08	1,39	1,68
	23	delay	684 271	21,52	17,50	566 424	39,93	12,20	631 197	**15,64**	9,46	1,08	1,38	1,85
	23	power	655 982	21,62	15,00	570 710	58,01	9,58	676 731	15,88	9,48	0,97	1,36	1,58
16) rs enconder [24]	6	area	13 883	3,00	6,90	**6 658**	3,19	2,76	7 269	3,09	2,77	1,91	0,97	2,49
	6	delay	16 004	2,83	7,81	8 553	2,51	3,01	7 880	2,93	2,77	2,03	0,97	2,82
	6	power	13 771	3,17	7,31	7 363	4,79	2,99	8 658	3,71	**2,63**	1,59	0,85	2,78
17) sasc [25]	4	area	8 697	2,23	4,70	**5 037**	3,44	2,17	5 449	2,6	2,01	1,6	0,86	2,34
	4	delay	9 139	**2,05**	4,84	5 828	2,56	2,37	5 608	2,44	2,11	1,63	0,84	2,29
	4	power	8 693	2,26	4,62	5 874	2,68	2,71	6 075	2,83	**1,93**	1,43	0,8	2,39
18) sha3 [26]	37	area	604 192	11,43	6,94	**418 803**	314,93	38,20	656 103	15,83	31,70	0,92	0,72	0,22
	37	delay	643 736	**9,95**	6,96	578 343	160,39	57,80	690 108	15,34	33,70	0,93	0,65	0,21
	37	power	556 241	10,18	**6,09**	509 170	474,79	46,50	690 722	16,21	35,60	0,81	0,63	0,17
19) simple spi [27]	8	area	10 860	3,53	1,67	**6 371**	3,28	**0,94**	7 147	2,49	**1,01**	1,52	1,42	1,65
	8	delay	11 139	**2,22**	1,69	7 906	2,98	**1,11**	7 757	2,33	**1,01**	1,44	0,95	1,67
	8	power	10 785	3,60	1,64	7 098	3,63	**0,98**	8 147	2,71	**1,04**	1,32	1,33	1,58
20) spi [28]	13	area	35 579	5,84	12,20	**24 327**	21,82	3,10	31 944	6,98	4,20	1,11	0,84	2,9
	13	delay	40 082	**4,65**	13,40	28 217	12,77	3,15	32 217	6,55	4,14	1,24	0,71	3,24
	13	power	35 781	5,75	11,60	29 035	9,63	**2,88**	35 043	7,56	4,42	1,02	0,76	2,62
21) ss pcm [29]	3	area	6 890	1,85	2,92	**4 020**	1,76	1,49	4 292	**1,6**	**1,42**	1,61	1,16	2,06
	3	delay	7 297	1,70	3,84	4 448	1,95	1,54	4 389	1,74	**1,45**	1,66	0,98	2,65
	3	power	6 788	1,88	2,72	3 877	2,56	1,45	4 542	**1,59**	**1,39**	1,49	1,18	1,96
22) systolic array [30]	12	area	8 000	5,20	1,16	**4 970**	4,54	0,71	6 281	4,18	**0,22**	1,27	1,24	5,18
	12	delay	9 434	**3,15**	1,48	6 594	3,52	0,92	7 007	4,5	**0,24**	1,35	0,7	6,27
	12	power	7 147	4,38	1,07	5 852	5,88	0,88	7 032	4,44	**0,24**	1,02	0,99	4,48
23) tiny gpu [31]	6	area	10 028	5,04	1,24	**6 993**	5,57	**1,02**	7 799	5,59	**1,01**	1,29	0,9	1,23
	6	delay	10 619	**4,11**	1,28	8 519	6,89	1,13	8 615	6,06	**1,01**	1,23	0,68	1,27
	6	power	9 941	4,91	1,19	7 311	5,74	**1,05**	8 836	5,63	**1,01**	1,13	0,87	1,18
24) tv80 [32]	21	area	75 689	10,64	7,25	**40 466**	15,66	2,04	57 982	10,4	**1,28**	1,31	1,02	5,66
	21	delay	87 644	**7,22**	8,22	62 783	12,9	2,36	65 120	9,6	**1,27**	1,35	0,75	6,47
	21	power	74 135	9,17	6,32	45 869	18,8	2,05	64 443	10,44	**1,29**	1,15	0,88	4,9
25) usb fdu [33]	26	area	70 015	6,49	1,90	**39 533**	13,86	1,15	57 881	10,27	**0,84**	1,21	0,63	2,27
	26	delay	74 871	**5,87**	1,89	57 265	9,97	1,29	67 092	10,52	**0,84**	1,12	0,56	2,25
	26	power	69 114	6,94	1,82	43 566	12,75	1,00	65 002	11,99	**0,84**	1,06	0,58	2,17
26) usb funct [34]	7	area	168 096	5,69	10,50	**124 320**	12,03	4,56	130 648	**1,93**	4,32	1,29	2,95	2,43
	7	delay	176 200	5,54	14,80	134 491	12,03	4,67	131 600	**1,95**	4,18	1,34	2,84	3,54
	7	power	165 832	5,61	9,99	132 196	12,03	4,67	144 397	**1,94**	**3,77**	1,15	2,89	2,65
27) usb phy [35]	4	area	7 492	2,05	2,89	**4 289**	2,28	**1,26**	4 592	1,8	**1,25**	1,63	1,14	2,31
	4	delay	7 646	1,91	3,26	4 429	**1,59**	**1,28**	4 882	1,77	**1,27**	1,57	1,08	2,57
	4	power	7 462	2,01	2,82	4 478	2,34	1,33	5 127	1,96	**1,20**	1,46	1,03	2,35

Design number/ Design name	clk per.	Obj.	Yosys (i)			Utopia EDA (ii)			Utopia+buffering (iii)			Ratio (1)/(2)		
			area, um2	delay, ns	power, mW	area, um2	delay, ns	power, mW	area, um2	delay, ns	power, mW	area	delay	po-wer
28) Verilog lfsr [36]	*6*	*area*	*7 134*	*2,79*	*2,09*	***4 220***	*2,07*	*1,36*	*4 354*	*2,14*	***1,37***	*1,64*	*1,30*	*1,53*
	6	*delay*	*9 229*	*2,20*	*1,86*	*5 137*	***1,95***	*1,60*	*4 619*	*2,14*	***1,43***	*2,00*	*1,03*	*1,30*
	6	*power*	*7 022*	*2,58*	*2,01*	*4 504*	*2,16*	*1,50*	*4 881*	*2,67*	*1,46*	*1,44*	*0,97*	*1,38*
29) vga lcd [37]	*32*	*area*	*1 325 678*	*5,30*	*34,60*	***677 945***	*215,63*	*34,90*	*710 511*	*4,41*	***24,40***	*1,87*	*1,20*	*1,42*
	32	*delay*	*1 429 710*	*5,30*	*35,80*	*887 415*	*215,56*	*35,50*	*926 257*	***4,22***	***24,70***	*1,54*	*1,26*	*1,45*
	32	*power*	*1 352 916*	*5,35*	*35,10*	*691 826*	*217,27*	*26,20*	*757 627*	*4,46*	***24,40***	*1,79*	*1,20*	*1,44*
30) wb conmax [38]	*12*	*area*	*343 904*	*4,90*	*42,70*	***146 993***	*8,26*	*8,88*	*226 382*	***4,25***	***5,22***	*1,52*	*1,15*	*8,18*
	12	*delay*	*365 484*	*4,97*	*50,00*	*195 299*	*10,01*	*27,30*	*259 342*	***4,32***	*5,87*	*1,41*	*1,15*	*8,52*
	12	*power*	*338 171*	*4,97*	*50,00*	*171 331*	*15,94*	*9,86*	*259 485*	*4,65*	***5,28***	*1,30*	*1,07*	*9,47*
31) wb dma [39]	*8*	*area*	*55 920*	*3,89*	*6,18*	*37 744*	*7,34*	*3,67*	*40 693*	*5,84*	*3,57*	*1,37*	*0,67*	*1,73*
	8	*delay*	*59 556*	***3,11***	*9,69*	*38 265*	*7,57*	*3,74*	*40 048*	*5,43*	*3,55*	*1,49*	*0,57*	*2,73*
	8	*power*	*55 941*	*4,09*	*8,92*	***31 560***	*19,13*	***3,51***	*44 748*	*6,18*	***3,53***	*1,25*	*0,66*	*2,53*

The main conclusions drawn from the table are as follows:

1. Utopia EDA with buffering outperforms Yosys in the majority of cases when area (30 out of 31 designs) or power (29 out of 31 designs) is the optimization objective. However, it underperforms Yosys for delay optimization, achieving only 14 wins out of 31 cases.
2. Utopia EDA without buffering generally outperforms Yosys in terms of absolute values, while the introduction of buffering increases area. At the same time, buffering enables significant improvements in other characteristics, while still maintaining area results better than those obtained with Yosys. Moreover, if these characteristics were evaluated after physical synthesis, the apparent advantage of Utopia EDA without buffering would likely turn into a disadvantage: physical design becomes impractical for circuits with more than approximately 10+ outputs of cells, and buffering resolves this issue already at the logic synthesis stage.
3. There exist several designs for which Yosys produces significantly better results (e.g., sha3). Such cases require deeper investigation and are a subject of future analysis.

5. Conclusion

We have developed an implementation of functional technology mapping within the Utopia EDA framework. In experiments conducted on a diverse set of RTL models from multiple application domains, we generated technology-mapped netlists using our approach, including variants with additional buffering. For comparison, netlists were also generated using Yosys, and their characteristics were evaluated using OpenLane. The experimental results show that, in the majority of cases, the proposed functional technology mapping implementation outperforms Yosys when area or power is the optimization objective. Future work will focus on improving performance when delay is the primary optimization target.

References

[1]. Mishchenko A., Cho S., Chatterjee S., and Brayton R. Combinational and sequential mapping with priority cuts. In Proc. ICCAD, 2007.

[2]. Mishchenko A. ABC. Available at: https://people.eecs.berkeley.edu/~alanmi/abc, accessed 01.11.2025.

[3]. Wolf C. Yosys Open SYnthesis Suite. Available at: https://github.com/YosysHQ/yosys, accessed 01.11.2025.
[4]. Efabless Corp. OpenLane. Available at: https://github.com/The-OpenROAD-Project/OpenLane, accessed 01.11.2025.
[5]. EPFL. Mockturtle. Available at: https://github.com/lsils/mockturtle, accessed 01.11.2025.
[6]. Calvino A. T., De Micheli G. Technology mapping using multioutput library cells. Proc. ICCAD, 2023.
[7]. Radi G., Calvino A. T., De Micheli G. In Medio Stat Virtus: Combining Boolean and Pattern Matching. ASP-DAC, 2024.
[8]. ISP RAS. Utopia EDA. Available at: https://gitlab.ispras.ru/mvg/utopia-eda, accessed 01.11.2025.
[9]. Spiking neural network accelerator. Available at: https://github.com/sfmth/OpenSpike, accessed 01.11.2025.
[10]. Discrete-Time FIR Multirate Filter. Available at: https://gitlab.ispras.ru/mvg/mvg-rtl/tt06-adpcm-compressor, accessed 01.11.2025.
[11]. WISHBONE AC97 Controller. Available at: https://github.com/ispras/hdl-benchmarks/tree/master/iwls05/opencores/rtl/ac97_ctrl, accessed 01.11.2025.
[12]. AES Cipher. Available at: https://github.com/ispras/hdl-benchmarks/tree/master/iwls05/opencores/rtl/aes_core, accessed 01.11.2025.
[13]. A DDR3 memory controller in Verilog for various FPGAs. Available at: https://github.com/ultraembedded/core_ddr3_controller, accessed 01.11.2025.
[14]. DES Module. Available at: https://github.com/ispras/hdl-benchmarks/tree/master/iwls05/opencores/rtl/des/area_opt, accessed 01.11.2025.
[15]. Tripple DES Module. Available at: https://github.com/ispras/hdl-benchmarks/tree/master/iwls05/opencores/rtl/des/perf_opt, accessed 01.11.2025.
[16]. Ethernet IP core. Available at: https://github.com/ispras/hdl-benchmarks/tree/master/iwls05/opencores/rtl/ethernet, accessed 01.11.2025.
[17]. WISHBONE rev B.2 compliant I2C Master controller. Available at: https://github.com/ispras/hdl-benchmarks/tree/master/iwls05/opencores/rtl/i2c, accessed 01.11.2025.
[18]. WISHBONE Memory Controller. Available at: https://github.com/ispras/hdl-benchmarks/tree/master/iwls05/opencores/rtl/mem_ctrl, accessed 01.11.2025.
[19]. An APB4-based PLIC. Available at: https://github.com/oscc-ip/plic, accessed 01.11.2025.
[20]. An AXI4-based PSRAM Controller. Available at: https://github.com/oscc-ip/psram, accessed 01.11.2025.
[21]. An AXI4-based VGA Controller. Available at: https://github.com/oscc-ip/vga, accessed 01.11.2025.
[22]. PCI bridge. Available at: https://github.com/ispras/hdl-benchmarks/tree/master/iwls05/opencores/rtl/pci, accessed 01.11.2025.
[23]. RaveNoC: A configurable HDL Network-On-Chip suitable for different MP applications. Available at: https://github.com/ aignacio/ravenoc, accessed 01.11.2025.
[24]. Reed Solomon Encoder and Decoder Digital IP. Available at: https://github.com/RedFlag2017/rs-codec, accessed 01.11.2025.
[25]. Simple Asynchronous Serial Comm. Device. Available at: https://github.com/ispras/hdl-benchmarks/tree/master/iwls05/opencores/rtl/sasc, accessed 01.11.2025.
[26]. SHA3 Accelerator. Available at: https://github.com/ucb-bar/sha3, accessed 01.11.2025.
[27]. MC68HC11E based SPI interface. Available at: https://github.com/ispras/hdl-benchmarks/tree/ master/iwls05/opencores/rtl/simple_spi, accessed 01.11.2025.
[28]. SPI IP core. Available at: https://github.com/ispras/hdl-benchmarks/tree/ master/iwls05/opencores/rtl/spi, accessed 01.11.2025.
[29]. PCM IO Slave Module. Available at: https://github.com/ispras/hdl-benchmarks/tree/ master/iwls05/opencores/rtl/ss_pcm, accessed 01.11.2025.
[30]. Systolic array in Verilog. Available at: https://github.com/Dazhuzhu-github/systolic-array, accessed 01.11.2025.
[31]. A minimal GPU design in Verilog. Available at: https://github.com/adam-maj/tiny-gpu, accessed 01.11.2025.
[32]. TV80 8-Bit Microprocessor Core. Available at: https://github.com/ispras/hdl-benchmarks/tree/master/iwls05/opencores/rtl/tv80, accessed 01.11.2025.
[33]. Full Speed USB DFU interface for FPGA and ASIC designs. Available at: https://github.com/ulixxe/usb_dfu, accessed 01.11.2025.
[34]. USB function core. Available at: https://github.com/ispras/hdl-benchmarks/tree/master/iwls05/opencores/rtl/usb_funct, accessed 01.11.2025.

[35]. USB 1.1 PHY. Available at: https://github.com/ispras/hdl-benchmarks/tree/master/iwls05/opencores/rtl/usb_phy, accessed 01.11.2025.
[36]. Fully parametrizable combinatorial parallel LFSR/CRC module. Available at: https://github.com/alexforencich/verilog-lfsr, accessed 01.11.2025.
[37]. WISHBONE rev.B2 compliant enhanced VGA/LCD Core. Available at: https://github.com/ispras/hdl-benchmarks/tree/master/iwls05/opencores/rtl/vga_lcd, accessed 01.11.2025.
[38]. WISHBONE Connection Matrix Top Level. Available at: https://github.com/ispras/hdl-benchmarks/tree/master/iwls05/opencores/rtl/wb_conmax, accessed 01.11.2025.
[39]. WISHBONE DMA Top Level. Available at: https://github.com/ispras/hdl-benchmarks/tree/master/iwls05/opencores/rtl/wb_dma, accessed 01.11.2025.

Информация об авторах / Information about authors

Михаил Михайлович ЧУПИЛКО – кандидат физико-математических наук, старший научный сотрудник отдела технологий программирования ИСП РАН, старший научный сотрудник РЭУ им. Г.В. Плеханова. Сфера научных интересов: логический синтез, разработка цифровой аппаратуры, высокоуровневый синтез, верификация RTL-моделей аппаратуры.

Mikhail Mikhaylovich CHUPILKO – Cand. Sci. (Phys.-Math.), senior researcher at Ivannikov Institute for System Programming of the RAS, and a senior researcher at Plekhanov Russian University of Economics. Research interests: logic synthesis, development of digital hardware, high-level synthesis, verification of RTL-models.

Александр Сергеевич КАМКИН – кандидат физико-математических наук, ведущий научный сотрудник отдела технологий программирования ИСП РАН, ведущий научный сотрудник РЭУ им. Г.В. Плеханова. Научные интересы: формальные методы, синтез и верификация цифровой аппаратуры, гетерогенные компьютерные системы.

Alexander Sergeevich KAMKIN – Cand. Sci. (Phys.-Math.), leading researcher at Software Engineering department of ISP RAS, leading researcher at Plekhanov Russian University of Economics. Research interests: formal methods, synthesis and verification of digital hardware, and heterogeneous computer systems.

Даниил Ренатович ГАРЯЕВ является студентом РЭУ им. Г.В. Плеханова по специальности «Прикладная математика и информатика в экономике» , а также лаборантом в Институте системного программирования им. В.П. Иванникова. Его научные интересы включают технологическое отображение и генетические алгоритмы.

Daniil Renatovich GARYAEV is a student of the «Applied Mathematics and Computer science in economics» program at the Plekhanov Russian University of Economics. His research interests include technology mapping and genetic algorithms.

Егор Сергеевич БЕЛИН является студентом направления «Информационная безопасность» в МИЭМ ВШЭ, а также лаборантом в Институте системного программирования им. В.П. Иванникова. Его научная деятельность связана с разработкой алгоритмов и средств технологического отображения.

Egor Sergeevich BELIN is a student of the «Information Security» program at Moscow Institute for Electronics and Mathematics of Higher School of Economics. His research activity is related to the development and implementation of algorithms of technology mapping.

Григорий Алексеевич МАЗОВ является студентом направления «Компьютерная безопасность» в МИЭМ ВШЭ. Его научная деятельность связана с разработкой алгоритмов и средств технологического отображения.

Grigory Aleskeevich MAZOV is a student of the «Computer Security» program at Moscow Institute for Electronics and Mathematics of Higher School of Economics. His research activity is related to the development and implementation of algorithms of technology mapping.

Владислав Сергеевич ШТРЕНЕВ является студентом направления «Компьютерная безопасность» в МИЭМ ВШЭ, а также лаборантом в Институте системного программирования им. В.П. Иванникова. Его научная деятельность связана с разработкой алгоритмов и средств технологического отображения.

Vladislav Sergeevich SHTRENEV is a student of the «Computer Security» program at Moscow Institute for Electronics and Mathematics of Higher School of Economics. His research activity is related to the development and implementation of algorithms of technology mapping.