A Comprehensive Set of Logic Synthesis and Optimization Examples

Petr Fišer, Jan Schmidt
Faculty of Information Technology, Czech Technical University in Prague
fiserp@fit.cvut.cz, schmidt@fit.cvut.cz

Abstract

In this paper we introduce a new set of example circuits, primarily intended for using in logic synthesis and optimization, mostly for testing and benchmarking purposes. Basically, the proposed set of circuits is a collection of former popular benchmark sets. By putting these circuits together, we have formed a more comprehensive, but unified and well-arranged set of example circuits, from which a user can select circuits (or the whole benchmarks) upon his wishes and needs. The collection comprises of several sets, which, even though they sometimes contain the same circuits, are customized to particular needs of the user. This paper documents the example set, together with origins of its parts, and statistics on the circuits are provided.

1 Introduction

Benchmarking is extremely important for evaluation of algorithms and tools. In the targeted research domain, numerous benchmark sets have been introduced by now, mostly intended for logic synthesis and optimization [1]-[8] and circuit-level testing [9]-[13]. Benchmarks for system-level testing [14], high-level synthesis [15], etc. have been proposed too; however, they are out of the scope of our purposes.

The mentioned benchmarks are typically presented in different circuit specification formats, which can make them difficult to process universally and uniformly. This was one of the main motivations for creating a new set of example circuits.

For purposes of the proposed circuit collection, we aim at circuits described at *gate-level* only and without hierarchy. Thus, all the circuits are synthesizable and can be processed by available logic synthesis and optimization tools, as well as by gate-level ATPGs (Automatic Test Patterns Generators).

One of the major aims was to present a set of circuits that can be used for credible benchmarking. Therefore, the circuits do not contain any apparent redundancies (in terms of dangling gates, go-through wires, constant outputs, unconnected components, and similar). Basically, only the "useful" logic is present in the circuits. Let us note that filtering out these features (or even complete circuits with these features) will not make the circuits (or the collection) simpler; the "useful" logic is retained.

The paper is structured as follows: the motivations for devising a new example set are summarized in Section 2. Then, the origins of circuits in the set are described in Section 3. Section 4 presents the transformations performed upon these circuits. Section 5 introduces the naming conventions of the circuits and the collection structure. Section 6 presents some statistics of these circuits and illustrative experimental results. Section 7 concludes the paper.

2 Motivation

The major motivations for introducing a new example set are summarized in this section. The motivations came from observations of researchers dealing with logic synthesis and optimization and testing, and they reflect their needs. They are as follows:

1) *To combine circuits from several different sets.* For purposes of extensive benchmarking, the sizes of available benchmark sets are insufficient, both in terms of the number of benchmark circuits and their size; some benchmark sets present only relatively small circuits, while some other only the "big" ones. A set, where a large scale of circuit sizes will be available, would be beneficial.
2) *To unify their description.* As mentioned above, specification formats of the different benchmark sets differ. Therefore, we present the circuits in two universal descriptions: the Berkeley Logic

Interchange Format (BLIF) [16] suitable for academic tools [17], [18], [19] and structural VHDL commonly accepted by commercial tools.

3) Some benchmarks contain hierarchical descriptions of circuits. This may not be always welcome, since some tools do not support hierarchy. Also, the aim of the proposed collection is to present extracted circuits' logic, not a complex hierarchy. Thus, the circuits are presented in their *flattened* form, with hierarchy dissolved.
4) For some purposes, only combinational circuits are required. Even though it is not difficult to "cut" flip-flops from the circuit, it introduces an additional effort. Thus, we present both the original sequential circuits and their combinational parts. The user can then decide which collection to use.
5) Some benchmark circuits consist of numerous separated (unconnected) components. The splitting may happen when combinational parts are extracted from sequential circuits; however, it is even the case of many combinational and sequential circuits in original benchmarks. In an extreme case, there appear constant outputs and just "go-through" connections between the circuit inputs and outputs. This fact may cause difficulties in circuit processing, and more importantly, make the experimental results misleading. Therefore, we have *split* the circuits into connected components.

3 Circuits Origins

Most of the circuits in the proposed collection are taken from different popular benchmark sets, some are generated artificially in a generic way, some have undocumented origins. The development of several benchmark sets included in the proposed set of examples is illustrated in Fig. 1. These, together with some other benchmark sets, will be shortly reviewed in this section.

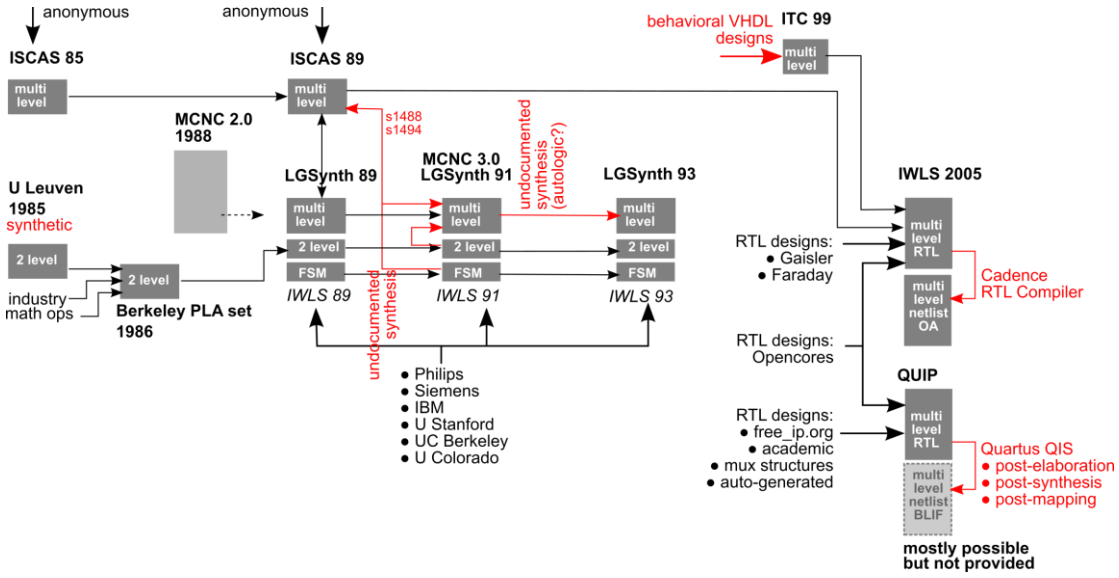


Figure 1: Development of the most popular benchmark sets

3.1 Espresso Examples

Probably the oldest logic optimization benchmark was a collection of PLA examples [1] distributed along with the SOP (Sum-of-Products) minimizer Espresso [20]. The circuits are combinational, described by two-level PLAs. These PLAs either come from practical designs, or they are randomly generated (e.g., the *ex1010* circuit).

3.2 MCNC Benchmarks

More general logic synthesis and optimization benchmarks come from Microelectronics Center of North California (MCNC) and are dated back to 1988 [2]. However, these benchmarks are no longer publicly available; only their remnants can be found, while most of the circuits are present in subsequent benchmark sets, like IWLS (see Figure 1). The benchmark consisted of both combinational and sequential circuits, two-level (PLA) descriptions are mixed with multi-level descriptions (BLIF [16]).

3.3 LGSynth'91 Benchmark

Most of the MCNC benchmark circuits were taken to the LGSynth'91 (sometimes called IWLS'91) benchmark set [3]. Here, two-level (PLAs) and multi-level descriptions are clearly distinguished, so we do that in our set too.

3.4 IWLS'93 Benchmark

The LGSynth'91 benchmark set was later extended by several other circuits [4], however, the resulting circuits are different. The BLIF files were synthesized into EDIF by an undocumented "Autologic" tool, which sometimes resulted in rather ridiculous and heavily redundant descriptions. Therefore, circuits from both sets ('91 and '93) are included in the proposed collection, since their descriptions are structurally different.

3.5 IWLS 2005 Benchmark

The most recent benchmark following the IWLS tradition is the IWLS 2005 set [5]. It contains the ISCAS and ITC'99 benchmarks (see the following subsections), together with big and practical circuits from OpenCores [21] and from industry.

3.6 ISCAS, ITC'99, and Illinois Testing Benchmarks

For purposes of benchmarking test generation tools (ATPGs) and generally presenting results obtained in the domain of circuits testing, three benchmark sets were proposed: ISCAS'85 [9] containing combinational circuits described by a netlist, ISCAS'89 [10] where several sequential circuits were added, and ITC'99 [11] containing a variety of practical sequential circuits. The set of testing benchmarks was yet supplemented by six synthesized circuits from Illinois University [13].

3.7 QUIP and Other Circuits from Altera

With the introduction of Altera's Quartus University Interface Program (QUIP) [22], a benchmark set was proposed too [7]. It contains several circuits intended for testing FPGA design flows. Even though the circuits are originally described at high level (VHDL and Verilog), they can be synthesized and converted to BLIF using Quartus Integrated Synthesis [22].

Altera recently proposed the "Advanced Synthesis Cookbook" [23], where there are numerous example designs. These, converted to BLIF after synthesis, are also included in the proposed set. However, they should be used with carefulness, since these circuits were not intended for benchmarking purposes.

3.8 LEKO/LEKU Example Circuits

Recently, Cong and Minkovich proposed several synthetic artificial designs that were used to test capabilities of logic synthesis for FPGAs [6]. Here an original, relatively small circuit was significantly enlarged by several equivalence preserving transformations. In the paper [6] they have shown that the enlarged circuits are far from optimum after logic optimization and synthesis. Therefore, they are very suitable for challenging the logic optimization process.

Two kinds of example circuits were presented – the LEKO and LEKU. The LEKO circuits are the original ones, not enlarged, thus with known optimum implementation complexity. The LEKU circuits are the enlarged ones, where the upper bounds on complexity are known – they equal to the complexities of LEKO circuits.

3.9 EPFL Benchmark Circuits

The most recent benchmark set comes from the Integrated Systems Laboratory at EPFL [8]. It consists of 23 natively combinational circuits aimed at challenging logic optimization tools and to provide a fair benchmarking platform; the obtained best 6-LUT synthesis results are available online and are regularly maintained. Recently, the set has been supplemented by several large two-level circuits described in PLA and several very big randomly generated multi-level netlists.

3.10 Generic Circuits

Apart from the standard benchmark circuits, the proposed set was extended by several generically generated circuits, particularly ripple-carry adders. Since they are regular and well understood

structures, they are suitable for testing scalability of synthesis tools, as well as their ability to efficiently treat XOR gates.

3.11 Other Circuits

Finally, there are some other circuits with not completely documented origin. Some of them come from OpenCores [21], some are manually constructed (e.g., the 74181 circuit), some were just released to provide supplementary material to some conference papers. Basically, they are not part of any mentioned benchmark set. However, we have decided to include them in the proposed collection, since they mostly represent practical and relatively large circuits.

4 Performed Transformations

In order to fulfill the requirements stated in Motivation, we have processed the benchmark sets as follows:

1) Circuits from all benchmark sets were converted to the BLIF format [16]. Particularly, PLA descriptions were converted by SIS [17], circuits described in the BENCH format [9] (namely those mentioned in Subsection 3.6) by our custom tool, circuits in VHDL by Quartus Integrated Synthesis [22]. Then all the circuits were accumulated, while their origin (original benchmark name) was indicated by the file name (see Section 5). Circuits that could not be converted to BLIF without a major loss of information, e.g., those described as finite-state machines (FSMs) in the KISS format, multi-valued descriptions (MV-PLA), etc., were removed from the set.
2) These circuits were read and written back by ABC [19]. By this, several actions have been done:
    a) circuits incorrectly converted to BLIF were removed from the set (this was especially the case of some Altera Cookbook circuits),
    b) extremely large circuits were removed, since ABC was not able to process them (this was the case of some large EPFL circuits),
    c) circuits with combinational feedback were removed,
    d) hierarchical descriptions were flattened,
    e) blackboxes [16] that appeared during the conversion to BLIF were removed,
    f) dangling gates were removed,
    g) circuits having external don't cares were removed, since ABC does not support them.
3) The ABC command 'short_names' was applied. As a result, a uniform signal naming convention was introduced to all circuits. Even though this may seem to be useless and even unwanted, it helps to avoid problems with signal names that are not supported by some other tools, or they are invalid identifiers in VHDL.
4) The circuits were processed by the 'sweep' command using ABC [19] or SIS [17]. By this, unnecessary buffers and inverters were removed, and constant signals propagated. Here we distinguished the circuits by their origin; some circuits were processed by ABC, some by SIS. The reason for this ambiguity was that ABC directly produces "complemented nodes", where found advantageous. This may completely hinder the structure (and purpose) of some circuits described by a PLA. Details on which tool was used for this transformation will be provided in Section 5.
5) The circuits were "cleaned" by our custom tool. Particularly:
    a) constant and void outputs were removed,
    b) primary outputs fed by a primary inputs only were removed, as well as primary inputs feeding primary outputs only.
6) Where applicable, each circuit was split into connected components. We have observed that this is quite a common case; many benchmark circuits actually comprise several unconnected blocks, which can be processed independently, without affecting the result anyhow. This is especially the case of sequential circuits from which combinational parts were extracted – see the following transformation.
7) Combinational parts of the circuits were extracted by the ABC command 'comb', making them combinational ones with pseudo-primary inputs/outputs. In this step, the circuit may break up into several unconnected components, which are extracted by the step mentioned above.
8) Small circuits were filtered out. Sometimes it happened that circuit parts obtained in Step 6) were too small, e.g., one gate only. Such circuits are thus useless for purpose of experimental evaluation. Next, the user may decide to use only circuits bigger than some threshold. Therefore, we offer sets with several size thresholds (5 and 50 gate equivalents).
9) When circuits from different benchmark sets are combined, one particular circuit may appear more than once in the mixed set. Therefore, we have performed three equivalence checking procedures:

a) File equivalence – exactly equivalent circuit descriptions were detected.
b) Structural equivalence – the circuits were processed by the ABC command 'strash', by which semi-canonical AIGs were produced, and the results were checked for equivalence. Therefore, structurally equivalent circuits were detected. Obviously, the a) class of equivalence is a subset of this class. Note that the 'strash' command actually introduces some structure. Thus, it may happen, e.g., that a circuit originally described in a PLA format, will be found equivalent to a multi-level netlist, when this netlist was directly produced from the PLA by ABC. However, multi-level, structurally significantly different descriptions will not be identified as equal.
c) Functional equivalence – functional equivalence of circuits was checked by the ABC command 'cec'. Thus, the result is independent of the structure. Obviously, the b) class of equivalence is a subset of this class.

The respective equivalences are recorded in text files and equivalence matrices (a symmetric rectangular matrix of dimensions [*n*, *n*], where *n* is the number of circuits in the whole set. A '1' entry at the position [*i*, *j*] indicates equivalence of the *i*-th and *j*-th circuit).

We have also created subsets of the original circuit set, where there is only one representative of each equivalence set present (alphabetically the first one). Thus, it is upon the user whether he wishes to use the complete set with possible duplicities, or a set with duplicities removed.

10) The circuits were converted to structural VHDL using our custom tool.

Let us note that the above transformations just present a summary of all actions performed. However, the proposed benchmark set comprises of results of almost all intermediate steps. Therefore, a user can freely decide what type of circuits to use, see Section 5.

5 The Proposed Circuit Collection

5.1 Circuit Naming Convention

As written above, the proposed set of circuits was constructed by accumulating circuits from several standard benchmark sets, while preserving the origins of circuits. For this purpose, the original benchmark set is indicated in the circuits' names as a prefix.

As described in step 6) in Section 4, the circuits were split into connected components, thus into several files. The ordinary number of the component is indicated by a suffix.

As a result, the circuits' naming convention is as follows:

*benchmark__name__*part*number*.*extension*

For example, the circuit *pair* coming from the LGSynth'91 benchmark set [3] was split into two components, which were named as (in the BLIF format):

LGSynth91__pair__part0.blif
LGSynth91__pair__part1.blif

In cases where the circuit was not split into components, the "__part" statement is omitted.

5.2 Circuits Origin and Count

The circuit origin, i.e., the standard benchmark set it comes from, is indicated by a prefix in its filename. The prefixes, together with references to the benchmark sets, their counts (in the `sweep` set, see Fig. 2), and the type of sweeping technique used ('sweep' by ABC or SIS) are listed alphabetically in Table 1:

Table 1: Circuit origins – filename prefixes

Prefix	Description	#	Sweep
Cookbook	Circuits from Altera Cookbook [23], Subsection 3.7	201	ABC
EPFL	EPFL benchmark [8], Subsection 3.9	29	ABC
Espresso	Espresso examples [1], Subsection 3.1	142	SIS
Generic	Artificially generated generic structures, Subsection 3.10	16	ABC
Illinois	Illinois circuits [13], Subsection 3.6	6	ABC
ISCAS	ISCAS'85 and ISCAS'89 [9], [10], Subsection 3.6	59	ABC

Prefix	Description	#	Sweep
ITC99	ITC'99 benchmark [12], Subsection 3.6	56	ABC
IWLS2005	IWLS'2005 benchmark [5], without ISCAS and ITC'99 circuits, Subsection 3.5	30	ABC
IWLS93	IWLS'93 benchmark [4], Subsection 3.4	141	SIS
LEKO	LEKO examples [6], Subsection 3.8	6	ABC
LEKU	LEKU examples [6], Subsection 3.8	2	ABC
LGSynth91-PLA	LGSynth'91 (IWLS'91) [3], Subsection 3.3, PLAs	40	SIS
LGSynth91	LGSynth'91 (IWLS'91) [3], Subsection 3.3	117	SIS
MCNC-Comb	MCNC [2], combinational circuits, Subsection 3.2	191	SIS
MCNC-Seq	MCNC benchmark [2], sequential circuits, Subsection 3.2	37	SIS
Other	Other, undocumented circuits, Subsection 3.11	28	ABC, SIS
QUIP	QUIP benchmark [7], Subsection 3.7	63	ABC

5.3 Collection Structure

The proposed collection of example circuits is structured hierarchically, in directories. Circuit files are stored in individual directories (compressed by 7z), while directory names represent respective transformations, as presented in Section 4. The structure is as follows:

```
2.0                                  ; the collection version, original circuit descriptions
└──Flat (1185)                       ; flattened hierarchy, processed by ABC step - 1), 2), 3)
    └──sweep (1164)                  ; processed by 'sweep' and cleaned - steps 4), 5)
        ├──Components (1836)         ; divided into connected components – step 6)
        │   ├──Size-5 (1346)         ; minimum size 5 gate equivalents – step 8)
        │   └──Size-50 (1103)        ; minimum size 50 gate equivalents – step 8)
        ├──Comb (1164)               ; combinational parts extracted – step 7)
        │   └──Components (31086)
        │       ├──Size-5 (2045)
        │       └──Size-50 (1508)
        ├──Unique-file (955)         ; only unique files – step 9a)
        │   ├──Comb (955)
        │   │   └──Components (30701)
        │   │       ├──Size-5 (1793)
        │   │       └──Size-50 (1316)
        │   └──Components (1535)
        │       ├──Size-5 (1094)
        │       └──Size-50 (911)
        ├──Unique-struct (944)       ; only structurally unique circuits – step 9b)
        │   ├──Comb (944)
        │   │   └──Components (30685)
        │   │       ├──Size-5 (1779)
        │   │       └──Size-50 (1312)
        │   └──Components (1519)
        │       ├──Size-5 (1080)
        │       └──Size-50 (907)
        └──Unique-funct (883)        ; only functionally unique circuits – step 9c)
            ├──Comb (883)
            │   └──Components (30588)
            │       ├──Size-5 (1719)
            │       └──Size-50 (1257)
            └──Components (1452)
                ├──Size-5 (1020)
                └──Size-50 (875)
```

Figure 2: Circuits collection structure (the steps reference to Section 4)

The numbers of circuits in the each set are shown in the figure too, in parentheses.
Each directory contains respective zipped BLIF and VHDL files, statistical data, and a short description of the set. Information on circuit equivalences (see Section 4, step 9) is present in the '`sweep`' directory.

6 Basic Statistical Properties of the Sets

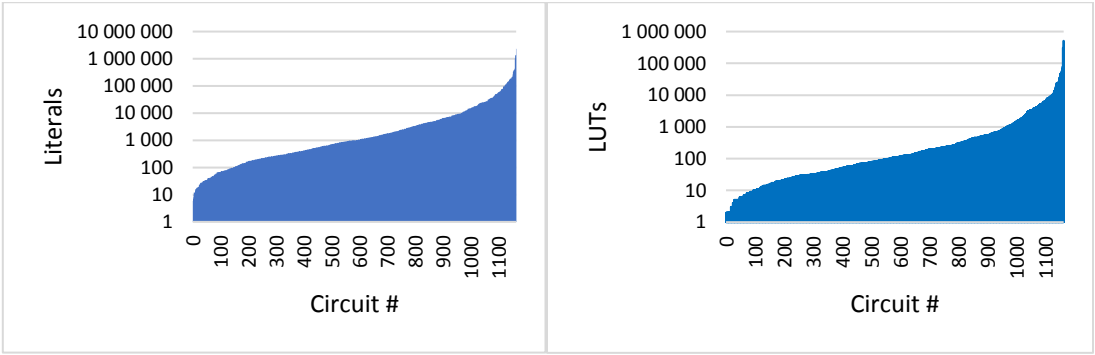
Some basic statistics on the proposed circuit sets will be provided here. Namely, the maximum and average numbers of inputs, outputs, nodes, flip-flops, literals, gate equivalents (GEs), levels (circuit depth), and the number of components will be provided for three basic sets: `sweep`, `sweep-components`, and `sweep-comb`. The "pruned" sets (minimum numbers of GEs, removed redundancy) are not listed here, since their properties are similar. Minimum numbers are not provided too, since they typically equal to 0 or 1.

Table 2: Basic statistical properties

	`sweep`		`sweep-components`		`sweep-comb`	
	Max.	Avg.	Max.	Avg.	Max.	Avg.
Inputs	8,018	81	8,017	58	185,040	821
Outputs	14,850	88	14,611	67	184,991	828
Nodes	1,167,052	6,493	1,167,052	4,837	1,167,052	5,753
Flip-flops	184,953	740	184,953	658	0	0
Literals	2,334,104	18,475	2,334,104	12,667	2,334,104	18,475
Levels	24,801	50	3,581	13	24,801	49
Components	245	2	1	1	4,469	27

Just to get an impression of the circuit sizes in the set, Fig. 3a shows the numbers of literals for all circuits in the `sweep` set. Each circuit is represented by a vertical line, while the circuits are sorted in ascending order by the number of literals. We can see that most of circuits are mid-sized ones (100–10,000 literals), however, there are also many small and some very large (> 1M literals) circuits.

These circuits were also optimized and mapped onto 4-input FPGA Look-up Tables (LUTs) by ABC, by a script 'strash; dch; if; mfs' iterated 20-times. The results, in terms of LUTs, are shown in Fig. 3b.



a) b)

Figure 3: Circuit sizes: a) literals in the original description, b) LUTs after synthesis

7 Conclusions

We have proposed a collection of example circuits for testing and possibly benchmarking logic synthesis and optimization tools. The collection is composed mostly of circuits from well-known standard benchmark sets. The reasons for introducing a new set of examples were to provide a large set of circuits, unify (standardize) their description, extract only "useful" logic from the circuits, and to offer the user several circuit sets to choose from, according to his/her needs, as circuits in these different sets were subject to different transformations. The circuits are available to the public at [24].

Acknowledgement

This work was partially supported by the grant GA16-05179S of the Czech Grant Agency, "Fault Tolerant and Attack-Resistant Architectures Based on Programmable Devices: Research of Interplay and Common Features" (2016-2018). Computational resources were provided by the MetaCentrum under the program LM2010005 and the CERIT-SC under the program Centre CERIT Scientific Cloud,

part of the Operational Program Research and Development for Innovations, Reg. no. CZ.1.05/3.2.00/08.0144.

References

[1] "Berkeley PLA Test Set Results", June, 1986.

[2] S. Yang, "Logic Synthesis and Optimization Benchmarks," Technical Report, MCNC, Dec. 1988, published at 1989 MCNC International Workshop on Logic Synthesis.

[3] S. Yang, "Logic Synthesis and Optimization Benchmarks User Guide Version 3.0," Technical Report 1991-IWLS-UG-Saeyang, MCNC.

[4] K. McElvain, "IWLS'93 Benchmark Set: Version 4.0," Distributed as a part of IWLS'93 benchmark set, May 1993.

[5] C. Albrecht, "IWLS 2005 Benchmarks," published at 2005 International Workshop on Logic Synthesis, June 2005.

[6] J. Cong and K. Minkovich, "Optimality study of logic synthesis for LUT-based FPGAs", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 2007, Vol. 26, No. 2, pp. 230–239.

[7] Altera, "Benchmark Designs for the Quartus University Interface Program (QUIP), Version 1.0," Altera, 2005.

[8] "The EPFL Combinational Benchmark Suite", Integrated Systems Laboratory LSI, École Polytechnique Fédérale De Lausanne, 2016, Available: http://lsi.epfl.ch/benchmarks.

[9] F. Brglez, H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortan," in Proc. of the International Symposium on Circuits and Systems, 1985, pp. 663-698.

[10] F. Brglez, D. Bryan, K. Kozminski, "Combinational Profiles of Sequential Benchmark Circuits," in Proc. of the Intl. Symposium of Circuits and Systems, 1989, pp. 1929-1934.

[11] F. Corno, M.S. Reorda, G. Squillero, "RT-level ITC'99 benchmarks and first ATPG results," in: Proc. of the IEEE Design and Test of Computers (2000), vol. 17, no. 3, pp. 44-53.

[12] CAD group, "ITC'99 Benchmarks (2nd release)". August 2009. Avalable: http://www.cad.polito.it/tools/itc99.html.

[13] V. Chickermane, J. Lee, and J. H. Patel, "A comparative study of design for testability methods using high-level and gate-level descriptions," Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, pp. 620-624, November 1992.

[14] E. J. Marinissen, V. Iyengar, and K. Chakrabarty, "A Set of Benchmarks for Modular Testing of SOCs," Proceedings of IEEE International Test Conference (ITC'02), Baltimore, MD, October 2002, pp. 519-528.

[15] N. Dutt and U.C. Irvine, "HLSW 92 benchmarks", 1992.

[16] University of California, Brekeley, "Berkeley logic interchange format (BLIF)," 2005.

[17] E.M. Sentovich et al., "SIS: A System for Sequential Circuit Synthesis," Electronics Research Laboratory Memorandum No. UCB/ERL M92/41, University of California, Berkeley, CA 94720, 1992, p. 52.

[18] M. Gao, Jie-Hong Jiang, Y. Jiang, Y. Li, S. Sinha, and R.K. Brayton, "MVSIS," in Notes of the International Workshop on Logic Synthesis, Tahoe City, June 2001.

[19] Berkeley Logic Synthesis and Verification Group, "ABC: A System for Sequential Synthesis and Verification" [Online]. Available: http://www.eecs.berkeley.edu/alanmi/abc/.

[20] R.K. Brayton et al., *Logic Minimization Algorithms for VLSI Synthesis*, Boston, MA, Kluwer Academic Publishers, 1984, 192 p.

[21] http://www.opencores.org

[22] Altera, "Synthesis Design Flows Using the Quartus University Interface Program (QUIP)," Altera, 2005.

[23] Altera, "Advanced Synthesis Cookbook", Altera, July 2011.

[24] http://ddd.fit.cvut.cz/prj/Bench_Examples/