The EPFL Combinational Benchmark Suite

Luca Amarú, Pierre-Emmanuel Gaillardon, Giovanni De Micheli
Integrated Systems Laboratory (LSI), EPFL, Switzerland

***Abstract*—In this paper, we present the EPFL combinational benchmark suite. We aim at completing existing benchmark suites by focusing only on *natively* combinational benchmarks. The EPFL ccombinational benchmark suite consists of 23 combinational circuits designed to challenge modern logic optimization tools. It is further divided into three parts. The first part includes 10 arithmetic benchmarks, e.g., square-root, hypotenuse, divisor, multiplier etc.. The second part consists of 10 random/control benchmarks, e.g., round-robin arbiter, lookahead XY router, alu control unit, memory controller etc.. The third part contains 3 very large circuits, featuring more than ten million gates each. All benchmarks have a moderate number of inputs/outputs ranging from few tens to about one thousand. The EPFL benchmark suite is available to the public and distributed in all Verilog, VHDL, BLIF and AIGER formats. In addition to providing the benchmarks, we keep track of the best optimization results, mapped into LUT-6, for size and depth metrics. Better logic implementations can be submitted online. After combinational equivalence checking tests, the best LUT-6 realizations will be included in the benchmark suite together with the author's name and affiliation.**

I. Introduction

The EDA community heavily relies on public benchmarks to evaluate the performance of academic and commercial design tools. Logic optimization and synthesis are core EDA applications where benchmarking is essential to develop effective methodologies. In this context, benchmarks can be broadly classified into two categories: combinational circuits and sequential circuits. Combinational circuit implements pure Boolean functions. Sequential circuit consists of combinational portions and memory elements. Sequential circuits can virtually describe any digital system. However, many academic optimization tools [1]–[7] only deal with combinational circuits because:

(i) The underlying optimization methodology is inherently intended for combinational logic.

(ii) Handling/splitting sequential circuits adds extra coding complexity in the software.

(iii) The optimization of a sequential circuit will eventually collapse into the optimization of its combinational portions.

On the other hand, tools dealing with sequential circuits are compatible with combinational circuits. With the aim of providing the best portability among existing synthesis and optimization tools, we focus on combinational benchmarks.

Existing combinational benchmark suites face one or more of the following issues:

(i) Large combinational circuits are derived from sequential circuits by removing registers and adding new inputs and outputs. This leads to a disproportionate number of I/O where tuning/testing optimization heuristics is inappropriate.

(ii) Natively combinational circuits are quite outdated. For example, the MCNC benchmark suite (1991) is 24 years old.

(iii) A benchmark intended behavior (original functionality) is often not provided or not discernible. This makes difficult drawing conclusions from the results.

(iv) Benchmarks provided in unrestricted HDL format are hard to read. General HDL parsers can be more complex than the optimization code itself.

(v) EDA vendors use customer designs as benchmarks to demonstrate performance of their products over their competition. However, customer designs are usually confidential and not available to the public.

(vi) Due to issues (i-v) researchers use custom benchmarks to test their tools. This makes difficult comparing to other approaches and again drawing conclusions from the results.

In this paper, we present the EPFL combinational benchmark suite [8]. On top of addressing the aforementioned issues, the EPFL benchmark suite has the following key features:

(i) All benchmarks are *natively* combinational.

(ii) Each benchmark is provided in all Verilog, VHDL, BLIF and AIGER formats. The HDL versions are written in a restricted format such that all assignments are simple AND-2/INV functions. The parsing complexity is minimized.

(iii) Three types of benchmarks are provided: arithmetic, random/control and very large (more than ten million gates).

(iv) Varying complexity of the benchmarks. Circuits with 10^i equivalent gates are available, with $i = 2, 3, 4, 5, 6, 7$. Different methods (exact or heuristic) can be tested on the most approriate circuit size.

(v) Each benchmark is accompanied by documentation about its functionality. By running logic simulation it is possible to reproduce the described functionality.

(vi) Researchers can submit their optimized circuits mapped into LUT-6 [8]. We will publish online the best version of each benchmark (size and depth metrics) together with the author's name and affiliation.

The remainder of this paper is organized as follows. Section II provides an historical background on logic benchmarks. Section III presents the EPFL combinational benchmark suite. Section IV shows simpele LUT-6 mapping experiments. Section V discusses the aim of the EPFL combinational benchmark suite initiative. Section VI concludes the paper.

II. Historical Background

The first set of combinational benchmark circuits was reported at the *International Symposium on Circuits and Systems* (ISCAS) in 1985 [9]. After four years, sequential circuits were added to ISCAS'85 generating the ISCAS'89 benchmark suite [10]. In 1991, these benchmarks plus others presented at past workshops and conferences were collected and distributed under the maintainance of the *Microelectronics Center of North Carolina* (MCNC) [11]. The MCNC suite was published in the same year at the *International Workshop on Logic Synthesis* (IWLS). Even though quite outdated, MCNC benchmarks are still popular in academic research.

After MCNC, a plethora of other specialized benchmark suites have been proposed, e.g., high level synthesis [12], physical design [13], testing [14], FPGA [15], etc.. However, none of them reached the same adoption level as MCNC.

In 2005, a new set of benchmarks for logic synthesis was presented at the IWLS workshop under the name of IWLS'05 benchmark suite [16]. It consisted of 84 designs collected from various websites (OpenCores, Faraday, etc.) and previous benchmark suites (MCNC, ITC, etc.). While the IWLS'05 suite provides state-of-the-art sequential circuits, it does not support natively combinational ones. Indeed, extracting the combinational portions of IWLS'05 circuits leads to a disproportionate number of I/O whose function is mostly unknown or not discernible.

In this work, we aim at completing the IWLS'05 suite by focusing on *natively* combinational benchmarks.

III. The EPFL Combinational Benchmark Suite

In this section, we present the EPFL combinational benchmark suite. It consists of 23 combinational circuits grouped in three parts. We start by presenting the first part which includes 10 arithmetic benchmars. Then, we show the second part containing 10 random/control benchmarks. Finally, we introduce the last part composed of 3 *More than ten Million* (MtM) gates circuits.

The EPFL combinational benchmark suite can be downloaded at [8] in Verilog, VHDL, BLIF and AIGER formats.

A. Arithmetic Benchmarks

The set of arithmetic benchmarks in the EPFL suite consists of 10 circuits representing complex arithmetic functions. They are obtained by a simple-minded (automated) mapping of arithmetic computational algorithms, such as square-root, logarithm, multiplication etc., into basic logic gates. The initial implementations are intentionally sub-optimal to test the ability of optimization tools. The arithmetic benchmarks come in different bit-widths to provide diversity in the implementation complexity. Table I shows their initial characteristics in terms of *And/Inverter Graph* (AIG) representation. In total, the arithmetic benchmarks count about 0.37M nodes and 36k levels. A detailed description for each benchmark follows.

1) Adder: This benchmark represents a standard 2-operand binary addition function:

$\{cOut, f\} = a + b$

where signals a, b and f are 128 bit wide. The signal $cOut$ is 1 bit wide.

TABLE I
Arithmetic Benchmarks

Benchmark name	Inputs	Outputs	AND nodes	Levels
Adder	256	129	1020	255
Barrel shifter	135	128	3336	12
Divisor	128	128	44762	4470
Hypotenuse	256	128	214335	24801
Log2	32	32	32060	444
Max	512	130	2865	287
Multiplier	128	128	27062	274
Sine	24	25	5416	225
Square-root	128	64	24618	5058
Square	64	128	18484	250
Total	1663	1020	373958	36076

In its initial AIG implementation, this benchmark counts 1020 nodes and 255 levels.

2) Barrel shifter: This benchmark is a barrel shifter. Its logic behavior is:

$result =$right_shift$(a, shift)$

where *a* and *result* are 128 bit wide signals, *shift* is a 7 bit wide signal.

In its initial AIG implementation, this benchmark counts 3336 nodes and 12 levels.

3) Divisor: This benchmark implements an (unsigned) integer division. All signals are 64 bit wide:

$quotient$ rem $remainder = a/b$

that can be equivalently rewritten as:

$a = b\cdot quotient + remainder$

In its initial AIG implementation, this benchmark counts 44762 nodes and 4470 levels.

4) Hypotenuse: This benchmark computes the length of the longest side of a right-angled triangle, i.e., the hypotenuse. Let a and b the length of the other two sides in the triangle. The hypotenuse can be computed using the Pythagorean theorem:

$hypotenuse = \sqrt{a^2 + b^2}$

All signals are 128 bit wide.

In its initial AIG implementation, this benchmark counts 214335 nodes and 24801 levels.

5) Log2: This benchmark represents a logarithm base 2 function:

$result = log_2(a)$

In this benchmark, all signals are 32 bit wide.

In its initial AIG implementation, this benchmark counts 32060 nodes and 444 levels.

6) Max: This benchmark computes the maximum among four numbers {in0, in1, in2, in3}. On top of returning the maximum value, it also gives the address (from 0 to 3) where such maximum was located.

$result = max\{in0, in1, in2, in3\}$

$address =$position$(result)$

Signals {in0, in1, in2, in3} and *result* are 128 bit wide. Instead, signal *address* is 2 bit wide.

In its initial AIG implementation, this benchmark counts 2865 nodes and 287 levels.

7) Multiplier: This benchmark is a standard 2-operand binary multiplier. Its function is:

$f = a \cdot b$

where a, b are 64 bit wide. The signal f is 128 bit wide.

In its initial AIG implementation, this benchmark counts 27062 nodes and 274 levels.

8) Sine: This benchmark represents the sine trigonometric function. The input angle is a, and its function is:

$sin = \sin(a)$

Signal a is 24 bit wide and sin is 25 bit wide.

In its initial AIG implementation, this benchmark counts 5416 nodes and 225 levels.

9) Square-root: This benchmark realizes a square-root function:

$asqrt = \sqrt{a}$

Signal a is 128 bit wide while signal $asqrt$ is 64 bit wide. The difference in bit width is due to (unsigned) integer truncation at the output.

In its initial AIG implementation, this benchmark counts 24618 nodes and 5058 levels.

10) Square: This benchmark is a square function:

$asquared = a^2$

Signal a is 64 bit wide while signal $asquared$ is 128 bit wide.

In its initial AIG implementation, this benchmark counts 18484 nodes and 250 levels.

B. Random/Control Benchmarks

The set of random/control benchmarks in the EPFL suite consists of various types of controllers, arbiters, routers, converters, decoders, voters and random functions. It contains 10 circuits mapped into simple gates from behavioral descriptions. Table II shows their initial characteristics in terms of *and/inverter graph* (AIG) representation. Also here, the initial implementations are intendedly unoptimized. In total, the random/control benchmarks count about 76k nodes and 0.6k levels. A detailed description for each benchmark follows.

TABLE II
RANDOM/CONTROL BENCHMARKS

Benchmark name	Inputs	Outputs	AND nodes	Levels
Round-robin arbiter	256	129	11839	87
Alu control unit	7	26	174	10
Coding-cavlc	10	11	693	16
Decoder	8	256	304	3
i2c controller	147	142	1342	20
Int to float converter	11	7	260	16
Memory controller	1204	1231	46836	114
Priority encoder	128	8	978	250
Lookahead XY router	60	30	257	54
Voter	1001	1	13758	70
Total	2832	1841	76441	640

1) Round-robin arbiter: This benchmark implements the combinational tasks involved in round robin arbitration. Round robin arbitration is a scheduling scheme which gives to each requestor its share of using a common resource for a limited time or data elements [17]. In this particular benchmark, priority and request signals are 128 bit wide and the grant signal is 128 bit wide. The request signal follows a one-hot priority vector encoding. There is also a *anyGrant* signal saying if any grant has been issued. It is 1 bit wide.

In its initial AIG implementation, this benchmark counts 11839 nodes and 87 levels.

2) Alu control unit: This benchmark is a simple alu control unit. It has various signals controlling alu operations, register destination, memory operations, jumps, etc. It has 7 inputs and 26 outputs.

In its initial AIG implementation, this benchmark counts 174 nodes and 10 levels.

3) Coding-cavlc: This benchmark is part of a *Context-adaptive variable-length coding* (CAVLC) video encoder for the H.264/MPEG-4 AVC format [18]. It contains look-up tables for coefficients, total zeros, trailing ones and other signals [18]. Given its nature, this benchmark is an example of random logic.

In its initial AIG implementation, this benchmark counts 693 nodes and 16 levels.

4) Decoder: This benchmark is a traditional decoder. In particular, it is an 8 to 256 decoder.

In its initial AIG implementation, this benchmark counts 304 nodes and 3 levels.

5) i2c controller: This benchmark implements the combinational tasks of an i2c communication controller. The full description is taken from OpenCores [19]. Due to its complex functionality, the precise I/O functionality is not provided.

In its initial AIG implementation, this benchmark counts 1342 nodes and 20 levels.

6) Int to float converter: This benchmark represents an integer to floating point format conversion. The input is an integer in binary format named B. B is 10 bit wide. The outputs are a mantissa signal M and an exponent signal E. M and E are 4 and 3 bit wide, respectively.

In its initial AIG implementation, this benchmark counts 260 nodes and 16 levels.

7) Memory controller: This benchmark implements a memory controller. Similarly to the i2c controller, it is taken from a full design available at OpenCores [20]. Due to its complex functionality, the precise I/O functionality is not provided.

In its initial AIG implementation, this benchmark counts 46836 nodes and 114 levels.

8) Priority encoder: This benchmark is a standard priority encoder. In particular, it is a 128 to 7 priority encoder. There is one additional output bit indicating whether or not the input data is valid.

In its initial AIG implementation, this benchmark counts 978 nodes and 250 levels.

9) Lookahead XY router: This benchmark represent a popular router in network-on-chips. It is a lookahead XY router to achieve low latency, high throughput communication in a network-on-chip environment. Details on the lookahead XY router are available at [21].

In its initial AIG implementation, this benchmark counts 257 nodes and 54 levels.

10) Voter: This benchmark is a voter circuit. In particular, it is a majority voter of 1001 variables.

In its initial AIG implementation, this benchmark counts 13758 nodes and 70 levels.

C. MtM Benchmarks

The *More than ten Million gates* (MtM) benchmarks are designed to challenge the size capacity of modern optimization tools. In the EPFL combinational benchmark suite there are three such MtM circuits. They are extracted from a set of random Boolean functions, generated with a custom computer program, using as selection metric the implementation complexity. Given their relatively small number of I/O, the MtM benchmarks may have a much smaller minimum implementation complexity. However, no optimization tool found yet a substantially smaller AIG implementation.

The three MtM benchmarks are named *sixteen*, *twenty* and *twentythree*. Their names indicate the number of million AND nodes in their initial AIGs. Table III shows their characteristics in terms of *and/inverter graph* (AIG) representation. In total, the MtM benchmarks count about 60M nodes and 0.48k levels.

TABLE III
MtM Benchmarks

Benchmark name	Inputs	Outputs	AND nodes	Levels
sixteen	117	50	16216836	140
twenty	137	60	20732893	162
twentythree	153	68	23339737	176
Total	407	178	60289466	478

IV. LUT-6 Mapping Experiments

In this section, we present LUT-6 mapping experiments for the EPFL combinational benchmark suite. Table IV shows the corresponding results. All experiments are performed using ABC academic tool [22], with the pure mapping command *if -K 6*. Obviously, better results were possible by including optimization scripts prior to the mapping step. However, finding the best LUT-6 realization for these benchmarks if left as challenge to researchers. Details on how to submit better implementations for one or more of the EPFL benchmarks are given in [8].

In our LUT-6 mapping experiments, we observe a coherent but scaled trend with respect to the initial AIG implementations. Arithmetic benchmarks are mapped into 80k LUT6 and 6.4k levels in total. Random/control benchmarks are mapped into 18k LUT6 and 0.1k levels in total. MtM benchmarks are mapped into 21M LUT6 and 1k levels in total. The average AIG/LUT-6 scaling factor for size ranges between 3 and 5. Considering depth, the average AIG/LUT-6 scaling factor ranges between 4 and 6.

TABLE IV
LUT-6 Mapping Experiments

Benchmark name	Inputs	Outputs	LUT-6 count	Levels
Arithmetic				
Adder	256	129	254	51
Barrel shifter	135	128	512	4
Divisor	128	128	9311	867
Hypotenuse	256	128	44635	4194
Log2	32	32	8008	77
Max	512	130	842	56
Multiplier	128	128	5913	53
Sine	24	25	1458	42
Square-root	128	64	5720	1033
Square	64	128	3985	50
Total	1663	1020	80638	6427
Random/Control				
Round-robin arbiter	256	129	2722	18
Alu control unit	7	26	29	2
Coding-cavlc	10	11	122	4
Decoder	8	256	287	2
i2c controller	147	142	365	4
Int to float converter	11	7	49	3
Memory controller	1204	1231	12096	25
Priority encoder	128	8	210	31
Lookahead XY router	60	30	89	7
Voter	1001	1	2691	16
Total	2832	1841	18660	112
MtM				
sixteen	117	50	5648909	29
twenty	137	60	7189658	33
twentythree	153	68	8246898	36
Total	407	178	21085465	98

V. Discussions

The EPFL benchmark suite is designed to test state-of-the-art optimization and synthesis tools. It consists of *natively* combinational circuits to guarantee the best portability between academic and commercial tools.

The EPFL benchmark suite fits a wide spectrum of optimization algorithms thanks to its variety of circuit types and complexity. For example, high-quality (time-consuming) methods suit small benchmarks such as *alu control unit* and *int2float converter* while fast methods (low-quality) suit large benchmarks such as *MtM* and *hypotenuse* benchmarks. Advanced synthesis tools with (auto)tunable quality/runtime tradeoff can be tested over the entire benchmark suite.

The ultimate goal of the EPFL benchmark suite is to define a new comparative standard for the logic optimization and synthesis community. We plan to achiave this goal by (i) providing a ready/easy to use set of benchmarks, (ii) keeping track of the best results and publishing them online (iii) giving a symbolic recognition/award to the authors of the best results.

We plan on extending the number and types of benchmark based on users' feedback.

VI. Conclusion

In this paper, we presented the EPFL combinational benchmark suite. It consists of 23 combinational circuits designed to challenge modern logic optimization tools. The benchmark suite is divided into *arithmetic, random/control* and *MtM* parts. The *arithmetic* part includes 10 benchmarks, e.g., square-root, hypotenuse, divisor, multiplier etc.. The *random/control* part consists of 10 benchmarks, e.g., round-robin arbiter, lookahead XY router, alu control unit, memory controller etc.. The *MtM* part contains 3 very large benchmarks, featuring more than ten million gates each. The EPFL benchmark suite can be downloaded at [8] in Verilog, VHDL, BLIF and AIGER formats. In addition to providing the benchmarks, we keep track of the best optimization results, mapped into LUT-6, for size and depth metrics. Better logic implementations can be submitted online. After combinational equivalence checking tests, the best circuits will be included in the benchmark suite together with the author's name and affiliation.

Acknowledgements

This research was supported by ERC-2009-AdG-246810.

References

[1] C. Yang, M. Ciesielski, *BDS: a BDD-based logic optimization system*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 21(7), 866-876, 2002.

[2] N. Vemuri, P. Kalla, R. Tessier. *BDD-based logic synthesis for LUT-based FPGAs* ACM Transactions on Design Automation of Electronic Systems (TODAES) 7.4 (2002): 501-525.

[3] T.S. Czajkowski, S.D. Brown, *Functionally linear decomposition and synthesis of logic circuits for FPGAs*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 27(12), 2236-2249, 2008.

[4] L. Amaru, P.-E. Gaillardon, G. De Micheli, *BDS-MAJ: A BDD-based logic synthesis tool exploiting majority logic decomposition*, Proc. DAC, 2013.

[5] L. Amaru, P.-E. Gaillardon, G. De Micheli, *Majority Inverter Graphs*, Proc. DAC, 2014.

[6] L. Amaru, P.-E. Gaillardon, G. De Micheli, *Boolean Logic Optimization in Majority Inverter Graphs*, Proc. DAC, 2015.

[7] N. Song, M. Perkowski *Minimization of Exclusive Sum of Products Expressions for Multi-Output Multiple-Valued Input, Incompletely Specified Functions*, IEEE Trans. on CAD, Vol. 15, No. 4, April 1996, pp. 385-395.

[8] EPFL Combinational Benchmark Suite: http://lsi.epfl.ch/benchmarks

[9] M. Hansen, H. Yalcin, J. Hayes, *Unveiling the ISCAS-85 Benchmarks: A Case Study in Reverse Engineering,* IEEE Design & Test of Computers, pp. 72-80, 1999.

[10] F. Brglez, D. Bryan, K. Kozminski, *Combinational Profiles of Sequential Benchmark Circuits*, Proc. 1989 Intl. Symposium on Circuits and Systems, May 1989.

[11] S. Yang, *Logic Synthesis and Optimization Benchmarks, Version 3.0,* Tech. Report, Microelectronics Center of North Carolina, 1991.

[12] High level synthesis benchmark – high level synthesis workshop https://filebox.ece.vt.edu/ mhsiao/hlsyn.html

[13] ISPD physical design contest suite http://www.ispd.cc/contests/14/web/benchmarks.html .

[14] *ITC benchmark suite*, http://www.cad.polito.it/downloads/tools/itc99.html .

[15] *OpenFPGA Working Groups*, http://www.openfpga.org /pages/WorkingGroups.aspx .

[16] IWLS 2005 Benchmarks. http://iwls.org/iwls2005/benchmarks.html

[17] E.S. Shin, V. J. Mooney, G. F. Riley. *Round-robin arbiter design and generation*, Proceedings of the 15th international symposium on System Synthesis. ACM, 2002.

[18] D. Marpe, H. Schwarz, T. Wiegand. *Context-based adaptive binary arithmetic coding in the H. 264/AVC video compression standard*, IEEE Transactions on Circuits and Systems for Video Technology, 13.7 (2003): 620-636.

[19] OpenCores, i2c controller project http://opencores.org/project,i2c

[20] OpenCores, memory controller project http://opencores.org/project,mem_ctrl

[21] A. Agarwal, C. Iskander, R. Shankar. *Survey of network on chip (noc) architectures & contributions*, Journal of engineering, Computing and Architecture 3.1 (2009): 21-27.

[22] ABC synthesis tool - available online at http://www.eecs.berkeley.edu/∼alanmi/abc/.