Small but Nasty Logic Synthesis Examples

Petr Fišer [*], Jan Schmidt [†]

Abstract

A method to find hard logic synthesis examples with known upper bound is presented. The circuits can be small and yet difficult to synthesize. Any area-related metric can be used in finding the circuits and testing synthesis tools. The hardness of the examples is robust with respect to the metric used and to minor alterations in the circuit.

1 Introduction

RTL description is presently the standard starting point of ASIC and FPGA industrial design. Logic synthesis, as the first step in RTL work flow, is believed to be a matured process, giving results reasonably close to optimum. Yet, Cong and Minkovich [1] published circuits with known optimal implementation (LEKO) or with known upper bound (LEKU), which are very hard for any synthesis process and causes it to yield results far from optimum; circuits obtained by the synthesis are up to 40x larger than expected.

While working on circuits testable on-line [2][3], we met circuits with similar properties. They are structurally different and much smaller. Both these properties permit us to try to understand the source of their hardness and to link them with research on symmetric functions [4].

We will briefly recapitulate the results of Cong and Minkowich, to establish a basis for comparison. Then, we are going to describe our LEKU circuits, with an algorithm for constructing candidates. By giving experimental results, we will demonstrate that the hardness is robust with respect to changes in the circuits. Finally, first hints possibly leading to understanding the hardness will be presented.

2 LEKU Circuits of Cong and Minkovich

Logic synthesis Examples with Known Optimal (LEKO) are constructed by replicating a relatively small circuit with n inputs and n outputs, given as a Boolean network with two-inputs nodes. Optimum mapping into 4-LUTs (4-input Look-up Tables) is known. After their multiple replication, there is a path from each input to each output. It has been proven that the optimum mapping of the entire circuit retains the mapping of the core circuit.

The resulting Boolean network can be used to evaluate the performance of LUT mappers against the proven optimum. The network can be also converted into a Sum-of-Product (SOP) description and used to evaluate any synthesis process capable of producing a 4-LUT mapping (Fig. 1). In this case, the proven mapping is only an upper bound, hence this type of evaluation circuit is referred to as Logic synthesis Examples with Known Upper bounds (LEKU).

When these LEKU benchmarks are synthesized using either open-source or commercial LUT mapping tools, the resulting number of LUTs is sometimes by two orders of magnitude larger than the known lower bound [1].

3 LEKU Circuits Based on Existing Benchmarks

During our research of a design of reliable self-checking circuits [2], there was a need to design a multiple-parity generator for a given circuit. The parity bits are produced by XOR-ing the

[*]Czech Technical University in Prague, fiserp@fel.cvut.cz
[†]Czech Technical University in Prague, schmidt@fel.cvut.cz

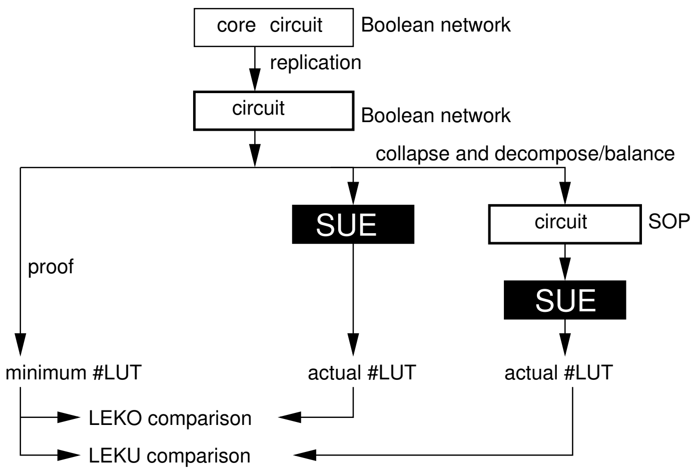

Figure 1: Circuit construction and data flow of Cong and Minkovich for a SUE (Synthesis under evaluation)

circuit's outputs. We initially supposed that the final parity generator logic would be smaller compared to the original circuit. The reason for the area reduction is straightforward: there is a chance that the appended XOR gates would be "soaked" into the original circuit by the synthesis, thus the logics would be reduced. However, we were rather surprised by the results; in some cases, the resulting parity generator circuit was almost 40-times bigger than the original circuit. Moreover, the size of the synthesized parity generator more than 25-times exceeded the size of a parity generator designed by manually appending XOR gates to its outputs. This sorrow fact gave us a hint for an investigation of the reason for such a failure of a standard synthesis process. The most apparent lack of quality was observed when XOR-ing all the circuit's outputs, in which case only one parity bit is obtained.

Based on the above-mentioned observations, our proposed method constructs a candidate benchmark circuit out of an arbitrary core circuit (Fig. 2) as follows:

Algorithm 1 *:*

```
1  Take any circuit with $m > 1$ outputs.
2  Construct a XOR tree with $m$ inputs.
3  Connect the tree to the outputs of the core circuit.
4  Collapse the whole circuit to obtain its two-level representation
```

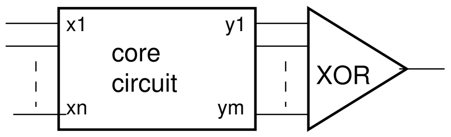

Figure 2: The proposed circuit construction

Collapsing the circuit with the XOR tree into its two-level representation is an essential step in the benchmark circuit generation. This way, the original (multi-level) structure of the circuit is completely obscured. Thus, the reconstruction (or refinement) of the former structure rests on the synthesis process only.

The candidate circuit has then to be measured as depicted in Fig. 3 with an area-related metric, such as the number of LUTs. Suitable circuits are distinguished by having the actual metric considerably bigger than the upper bound.

The number of 4-input LUTs has been chosen as a circuit's complexity metrics (also in correspondence with [1]), so that the upper bound could be easily determined: the minimum number of 4-LUTs needed to construct the XOR tree is $\lceil (m-1)/3 \rceil$, where m is the number of the original circuit's outputs. For example, the `alu1` MCNC benchmark circuit having 8 outputs may be implemented by 8 4-LUTs. Then, the upper bound of the complexity of the 1-parity generator circuit would be 8+3 = 11 4-LUTs.

4 Experimental Results

4.1 Comprehensive Exploration

Fig. 4 summarizes the results of an extensive measurement on the MCNC91 benchmarks [5] using the number of 4-input LUTs as a metric. It plots the ratio of the actual size to the upper bound (the nastiness factor) as a function of circuit size, measured by its upper bound. We used three different synthesis processes (SUEs):

- SIS [6] under the script recommended for LUT synthesis
- SIS under the `script_rugged` script, followed by `tech_decomp -a 4`, to obtain a network of 4-input nodes (i.e., LUTs as well)
- the LUT synthesis in ABC [7].

Each line in the plot represents one MCNC benchmark circuit. If the nastiness factor is below 1, the circuit size is reduced when its outputs are XOR-ed. Circuits having the factor greater than 1 are good candidates for a synthesis example. We observe the following:

There are hard-to-design circuits of all sizes. The smallest of them is `alu1`*, which also exhibits the greatest nastiness factor.*

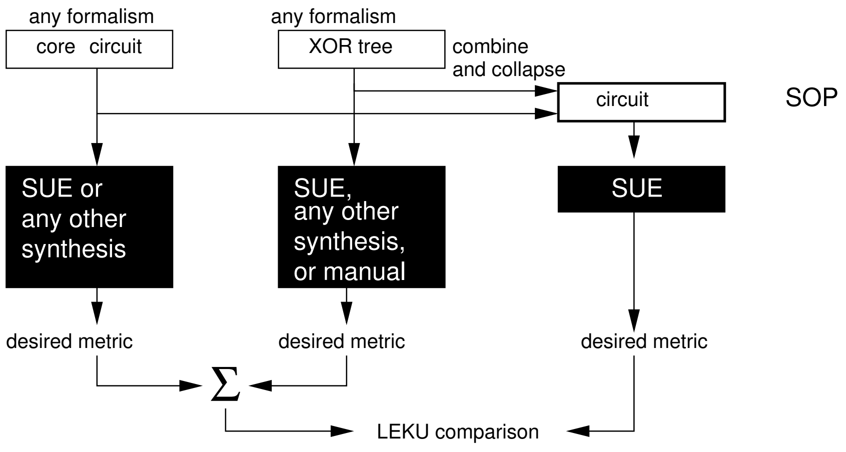

Figure 3: Proposed circuit construction and data flow for a SUE (Synthesis under evaluation)

In this measurement, we used the average number of LUTs as our metrics, together with the span of the number of LUTs.

4.2 "Nasty" And "Nice" Circuits

Experimental results obtained by several selected MCNC benchmarks are presented in Table 1. The "nasty" benchmarks are shown in the upper part of the table, while the "nice" ones are shown below.

The above-mentioned three synthesis processes have been applied to these circuits. After the circuit name, the theoretical lower bound of the number of 4-LUTs, computed as stated in Section 3, is shown. Then the numbers of 4-LUTs obtained by the circuit collapsing and a subsequent synthesis by SIS (LUT synthesis script), SIS using a `script_rugged` followed by `techdecomp -a 4` and ABC is shown in the following columns. The respective nastiness factors are indicated in brackets. The benchmarks are sorted by their nastiness factors.

It can be seen that the "nastiest" benchmark is `alu1`, with almost 26-times bigger implementation after the synthesis by SIS. The second SIS optimization script `script_rugged` followed by `techdecomp -a 4` always yields worse results than the suggested one. This gives us an important clue:

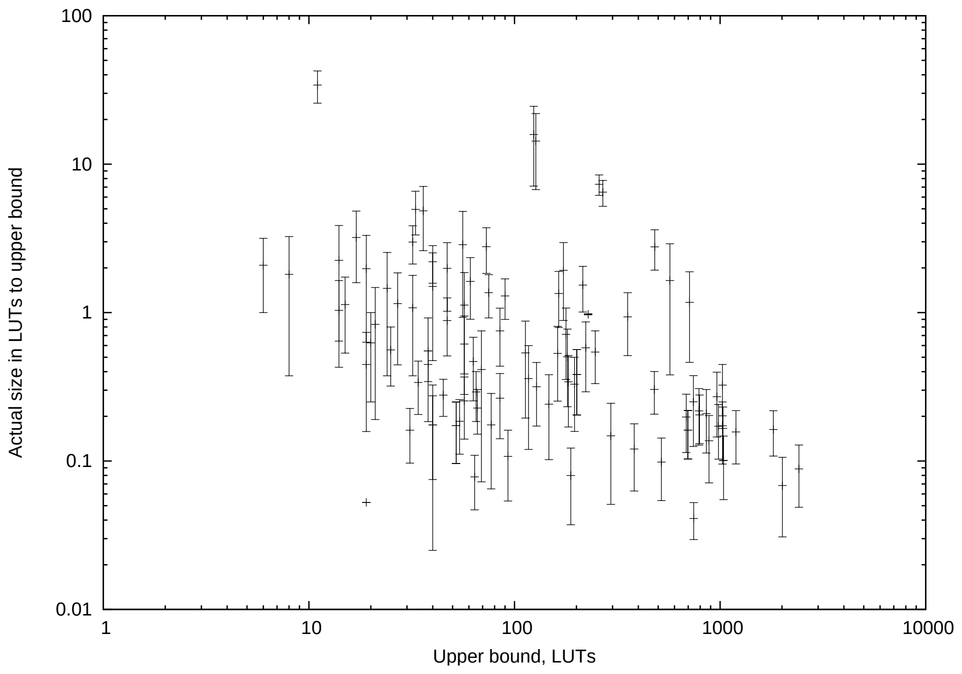

Figure 4: Measured results, MCNC benchmarks, three SUE

The problem with synthesizing these benchmark circuits is not caused by poor LUT-mapping. On the contrary, it is a general problem for any synthesis process.

Name	Theoretical lower bound	SIS LUTs	SIS LUTs (2)	ABC LUTs
alu1	11	283 (25.78)	467 (42.45)	444 (40.36)
misex3c	258	1590 (6.16)	1753 (6.79)	2182 (8.46)
alu3	33	110 (3.33)	217 (6.58)	139 (4.21)
alu2	36	94 (2.61)	255 (7.08)	113 (3.14)
b12	32	68 (2.13)	123 (3.84)	81 (2.53)
t1	73	142 (1.95)	272 (3.73)	134 (1.84)
alu4	481	929 (1.93)	1330 (2.77)	1738 (3.61)
t4	17	27 (1.59)	82 (4.82)	28 (1.65)
mp2d	40	63 (1.58)	113 (2.83)	70 (1.75)
e64	744	23 (0.03)	39 (0.05)	22 (0.03)
prom1	2010	62 (0.03)	213 (0.11)	125 (0.06)
pope	188	7 (0.04)	23 (0.12)	14 (0.07)
bw	64	3 (0.05)	7 (0.11)	6 (0.09)
mainpla	2417	118 (0.05)	310 (0.13)	142 (0.06)
lin	294	15 (0.05)	72 (0.24)	30 (0.10)
al2	93	5 (0.05)	15 (0.16)	8 (0.09)
exps	518	28 (0.05)	74 (0.14)	36 (0.70)
opa	382	24 (0.06)	68 (0.18)	27 (0.7)

Table 1: MCNC benchmarks ordered by nastiness, with LUT numbers as metrics

4.3 A Continuum between the Core and the Complete Circuit

It can be objected that the difficulty of the circuits is a result of random interaction between the core circuit and the XOR tree and that a small change in either of them will make the circuit ordinary. We tested this conjecture by splitting the outputs into groups and constructing a separate XOR tree for each group. In particular, we have varied the number of parity bits from 1 (which is our benchmarks case) to m (which corresponds to an original benchmark, without any parity XORs - the *core*). We have used two methods to produce the parity bits:

- A random assignment. The original circuit's outputs are distributed among the XOR trees at random.
- A more sophisticated technique based on the analysis of the circuit [3].

The results for four selected MCNC [5] benchmarks are shown in Fig. 5. The numbers of parity bits are indicated on the x-axis (starting from 1 to the number of the circuit's outputs), the y-axis shows the number of LUTs after the synthesis by SIS. The two curves correspond to the two grouping algorithms. Naturally the curves meet at their endpoints, where no optimization of the output grouping can be done. The upper curves correspond to the random grouping approach (yielding more LUTs). The values were obtained by averaging the results obtained by 500 runs of the whole synthesis process. For all circuits tested, the sequences between these limit points are monotonous, like as depicted in Fig. 5. Benchmarks `alu1` and `alu2` are typical "nasty" circuits. Their size rapidly increases with decreasing the number of parity bits. On the contrary, the `5xp1` and `duke2` are the "nice" ones, where appending XORs to their outputs yields less logics, as expected.

The monotonicity demonstrates that the nastiness of the circuit is robust with respect to circuit alteration. Moreover, it opens up a way to scale the nastiness.

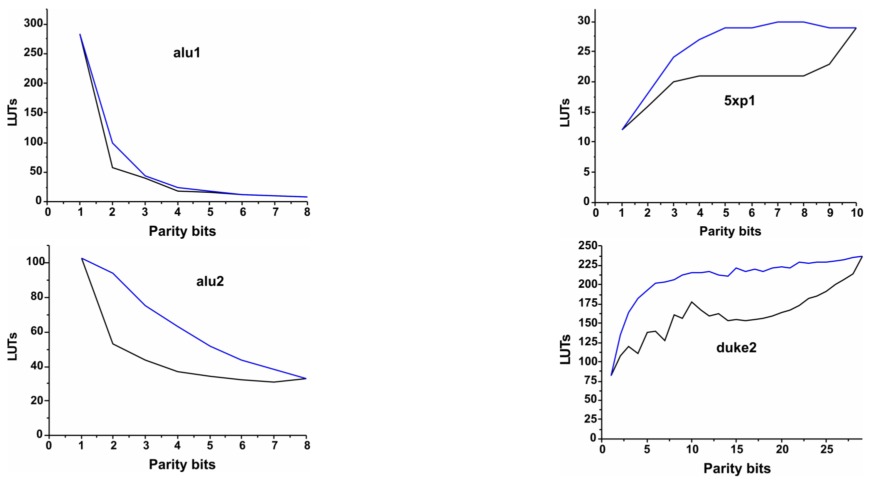

Figure 5: Variable number of XOR outputs

4.4 On Using Different Complexity Metrics

In previous measurement, we used the average number of LUTs as our metrics. Table 2 shows what happens when we change it to the average factored form number of literals, a radically different metrics. The ordering is the same by both metrics. This parallel continues – although

name	nastiness by LUT#	nastiness by factored form
alu1	25.78	8.12
misex3c	6.16	2.50
alu3	3.33	1.87
alu2	2.61	1.78
b12	2.13	1.03
t1	1.95	0.81

Table 2: MCNC benchmarks ordered by nastiness, with LUT number and factored form size as metrics

not precisely – for several more benchmarks, well into the region where the nastiness factor ceases to be interesting. We observe:

The nastiness is robust with respect to choice of size-related metrics.

5 `alu1` in Detail

As we stated before, the new circuits are fairly small. This enables us to gain some insight by reverse engineering and structure alterations of the core circuit. `alu1` is one of the smallest and yet nastiest circuits. Moreover, its name seems to suggest the presence of iterative structures and handcrafted arithmetic circuitry. Figure 6 is a detailed schematic of one possible implementation of the original PLA description. The core circuit is composed of four subcircuits, which all share

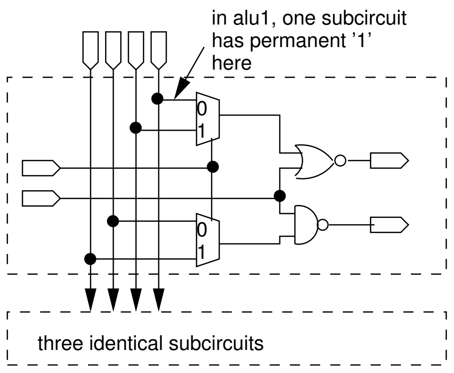

Figure 6: Alu1 schematic

four of the inputs. Each other input is processed by only one respective subcircuit. Three of the four subcircuits are identical, the fourth one is simplified (in the PLA form, one term is missing).

Figure 7 outlines our experimentation. Firstly, we tried to determine if the irregularity has any significance. The regularized core circuit, `alu1RG`, showed almost identical complexity results when augmented with the XOR tree (Table 3). This encouraged us to derive a small circuit, `alu1HFRG`, with only subcircuits from Figure 6. Even this toy example was not synthesized optimally. With circuits bigger than alu1RG, we ran into difficulties with the synthesis tools.

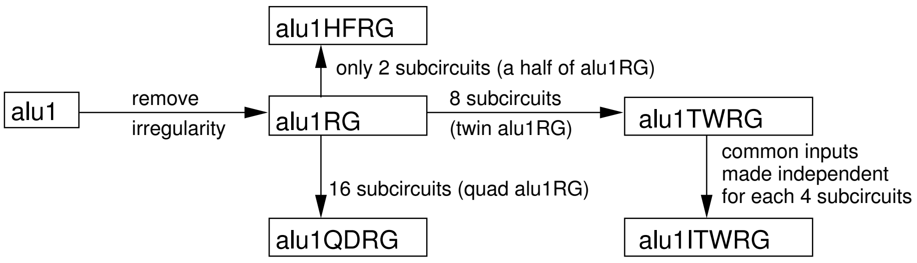

Figure 7: `alu1` derivatives

Name	Theoretical lower bound	SIS LUTs	ABC LUTs
`alu1`	11	283 (25.78)	444 (40.36)
`alu1RG`	11	280 (25.45)	491 (44.64)
`alu1HFRG`	5	9 (1.80)	28 (5.60)
`alu1TWRG`	21	fail	7826 (372,67)
`alu1QDRG`	43	fail	fail
`alu1ITWRG`	21	fail	1088 (51.81)

Table 3: The nastiness of `alu1` derivatives

The collapsing processes of SIS and MVSIS either produced normal forms of enormous size (up to 357 000 terms in the case of `alu1TWRG` and MVSIS), or failed to collapse the circuit (`alu1QDRG`). It is problematic to distinguish fails caused by the intermediate form and by the nastiness of the circuit itself. When produced, the normal form size differed greatly between SIS and MVSIS, which usually does not occur with “nice” circuit and which had a significant impact on synthesis results. Therefore, the results in Table 3 are obtained by “best effort”.

Nevertheless, `alu1TWRG` had record nastiness. The scatter points in Figure 4 seem to be contained within a convex body, which suggests that large nasty circuits may be sparse; `alu1TWRG` contradicts this.

To test the influence of reconvergent fan-out, we cut the four common inputs of `alu1TWRG` and gave each half of the circuit a separate set of four inputs in `alu1ITWRG`. Although the normal form was bigger than that of `alu1TWRG`, the nastiness dropped somewhat. It is still to be determined whether this is significant.

Replication (even partial replication) of nasty circuits can lead to bigger nasty circuits. Reconvergent fan-out can make the circuits harder. A better form to communicate solely the function (not the structure) of the circuits shall be found.

6 Related Research

The paper [4] by Wang and Dietmeyer, although aimed at synthesis improvement, brings observations that are probably relevant to our circuits. They construct an evaluation circuit by a method close to ours. They compose a MCNC benchmark, namely `f51m`, with the symmetrical circuit $S^8_{\{4\}}$. (In this notation, the XOR tree would be $S^8_{\{1,3,5,7\}}$). The purpose of this circuit is to demonstrate the advantages of a synthesis process that can handle *near symmetry* in circuits well. They also state that common algebraic decomposition methods perform poorly not only on symmetric functions but also on functions exhibiting near symmetry, moreover, that symmetry is not handled well in two-level minimization. These assertions, although intuitive, are not supported in the paper. Nevertheless, they give a possible explanation of the phenomena observed:

- Symmetric functions have logarithmic depth (and hence, linear circuit size).
- They are not handled well by contemporary algorithms.
- Near symmetry is in these respects similar to symmetry.
- Therefore, circuits with near symmetry tend to have small implementation, which is hard to find by contemporary algorithms.

So far, our experiments did not confirm this hypothesis.

7 Conclusions & Future Work

We propose a novel methodology to design example circuits that are difficult to be handled by a standard synthesis process. They are constructed by appending XOR trees to the outputs of a given circuit. These circuits, even though they are originally small, yield excessively large area after synthesis.

We have shown that the success of the synthesis monotonously depends on the number of parity bits generated. We have experimentally discovered an indispensable set of “nasty” core circuits, for which the complexity of the resulting complete circuit grows with the number of XOR sections appended. On the other hand, other circuits behave differently; the complexity of the XOR-ed circuit decreases with the number of its outputs (i.e. parity bits). This is the case one would expect.

Our ultimate goal is to understand the reasons why synthesis behave so differently in the cases described. The first conjecture to be verified is the influence of symmetry introduced into the circuit by the appended XOR gates, as suggested in [4].

References

[1] J. Cong and K. Minkovich, ”Optimality study of logic synthesis for LUT-based FPGAs” (2007). IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 26 (2), pp. 230–239. Postprint available free at: http://repositories.cdlib.org/postprints/2376

[2] P. Kubalík, P. Fišer and H. Kubátová, "Fault Tolerant System Design Method Based on Self-Checking Circuits", Proc. 12th International On-Line Testing Symposium 2006 (IOLTS'06), Lake of Como, Italy, July 10-12, 2006, pp. 185-186

[3] P. Fišer, P. Kubalík and H. Kubátová, "An Efficient Multiple-Parity Generator Design for On-Line Testing on FPGA", Proc. 11th Euromicro Conference on Digital Systems Design (DSD'08), Parma (Italy), 3. - 5.9.2008 (submitted)

[4] F. Wang and D. L. Dietmeyer, "Exploiting Near Symmetry in Multilevel Logic Synthesis" (1998). IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 17 (9), pp. 772–781.

[5] S. Yang, "Logic Synthesis and Optimization Benchmarks User Guide", Technical Report 1991-IWLS-UG-Saeyang, MCNC, Research Triangle Park, NC, January, 1991

[6] E. Sentovitch, K. Singh et al., "SIS: A System for sequential circuit synthesis", Univ. California, Berkeley, Tech. Rep., UCB/ERL M92/41, May 1992

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