On Identification of XOR Gates in AIGs

Ivo Háleček, Petr Fišer, and Jan Schmidt
Dept. of Digital Design, Faculty of Information Technology
Czech Technical University in Prague, Czech Republic
halecivo@fit.cvut.cz; fiserp@fit.cvut.cz; schmidt@fit.cvut.cz

1. Introduction

Logic synthesis and optimization had become a well-established and matured process in late 1980's. Since that time, different decomposition and optimization algorithms have been proposed [1], [2] and academic tools implementing these algorithms were released [3], [4]. Here, the internal representation of a processed circuit is typically a Boolean network, admitting arbitrary node functions. Technology library gates appear later, in the technology mapping phase [1], [2].

As a successor, algorithms based on And-Inverter-Graphs (AIGs) appeared [5], [6] and were implemented in the present academic state-of-the-art logic synthesis tool ABC [7]. The internal representation is a directed acyclic graph with nodes representing 2-input AND operators, while the edges may be negated. Such a representation is simple and scalable, and leads to simple algorithms.

Despite of unquestionable advantages of AIG-based algorithms, there appeared hints recently, that contemporary logic synthesis does not perform well in some cases [8], [9]; results orders of magnitude bigger than expected were produced. One of the identified reasons of the failures is the inability of the algorithms to produce results of *any structure*, particularly, XOR-intensive structure [10], [11].

Essentially, employing a new type of decomposition in a standard synthesis flow cannot ultimately solve the problem; the synthesis must be *adapted* to be able to perform more complex operations and natively support more complex transformations [11]. Even the most recent research in logic synthesis turns back from simple representations to more complex ones [12], [13].

The aim of this work is to extend the concept of AIGs to natively support XOR gates, i.e., to introduce a novel internal representation, Xor-And-Inverter-Graphs (XAIGs), and devise algorithms working upon it. XAIGs represent an *orthogonal* approach to Majority-Inverter-Graphs (MIGs), where the ANDs in AIG are substituted by majority-of-three functions [13]. In contrast, XOR is *not monotonic*, and XAIGs cover *all* relevant classes of NPN equivalence, which may lead to different areas of efficient application.

In this paper we present a preliminary work, where XOR structures are identified in an AIG. We study how structural hashing [6] in ABC [7] handles XOR gates coming from an already mapped netlist, whether they are structurally retained and whether new XOR structures can be identified in the AIG.

2. Identification of XORs in AIGs

To identify XOR gates in an AIG, AND-nodes are scanned recursively for XOR patterns. As seen in Figure 1, a XOR pattern consists of two input nodes, two or three inner nodes and one end node, which can be either inverted or not.

Once a pattern is found, fan-outs of the inner AND-nodes are checked. If no fan-out leading outside the XOR is found, the XOR is marked as "fanout-free". Nodes not marked as fanout-free are excluded from the total XOR count.

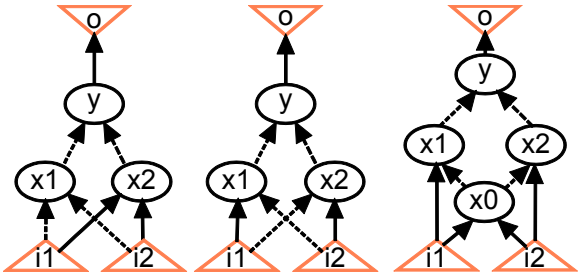

Figure 1. XOR structures in AIG

An example of an AIG of a 2-bit adder optimized and mapped by ABC is shown in Figure 2. The four identified XOR structures are encircled; however, only three of them can be used as gates, since the XOR structure rooted in node 22 is not fanout-free. However, its presence can be possibly exploited, provided that other nodes will provide the signals for nodes 10 and 12.

3. Experimental Results

We have performed the experiments using over 400 circuits from different benchmark sets [14], [15], [16], [17]. The ABC tool [7] was used to optimize and map the circuits into 2-input gates (AND, NAND, OR, NOR, XOR, XNOR) by a sequence of commands "strash; dch; map; mfs". The number of XOR gates in the resulting BLIF file was then taken as the first measure.

Next, the resulting gate-level description was converted back to an AIG by the ABC "strash" command, yielding a structurally hashed AIG [6]. XOR structures in the resulting AIGs were then identified, see Section II.

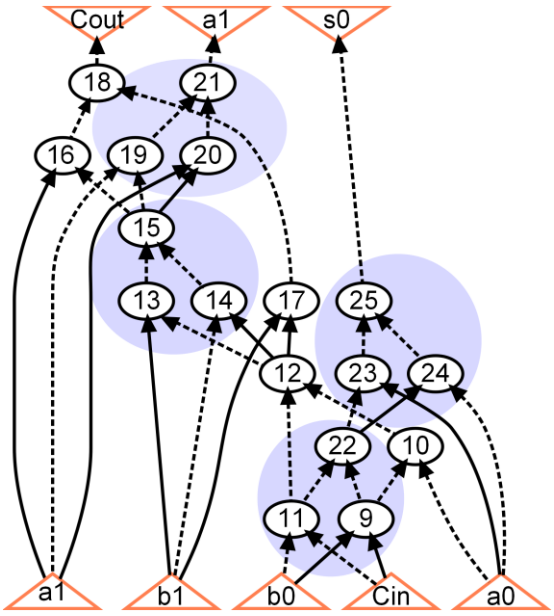

Figure 2. Identification of XORs in an AIG of a 2-input adder

Some representative results are in TABLE I. After the benchmark name, the number of gates in the mapped netlist and the number of AIG nodes after structural hashing are given. The column "XORs" indicates the number of XOR gates in the mapped netlist, "FF XORs" gives the counts of fanout-free XORs identified by our tool. Finally, the number of all identified XOR structures is shown in the "All XORs" column.

In several circuits, some XORs present in the mapped netlist have disappeared during the AIG construction (see, e.g., am2910). On the other hand, new XOR structures appeared in the AIGs in some cases (see s38417). Moreover, potential new XOR gates have been identified in most of cases (see the last column).

4. Conclusions & Future Work

An algorithm to detect XOR structures in AIGs was presented. Experimental results show that most of XORs present in the original mapped netlist are preserved in AIG. Moreover, newly formed XORs were discovered.

Our immediate future work will be focused on possibilities of exploiting XOR gates identified in the AIG in the optimization process and technology mapping. Next, we are going to introduce the XAIG structure, where 2-input XOR gates will be represented as a single node, in contrast to three nodes in AIGs. Finally, synthesis and optimization algorithms operating upon this structure will be devised.

Acknowledgement

This research has been in part supported by CTU grant SGS15/119/OHK3/1T/18.

TABLE I. EXPERIMENTAL RESULTS

Name	Gates	AIG nodes	XORs	FF XORs	All XORs
am2910	702	748	13	2	14
bca	2633	2449	0	0	13
bigkey	3527	3547	1	0	112
c1355	194	438	108	108	108
dalu	850	902	37	36	41
i10	1750	1866	86	84	110
parity	15	46	15	15	15
pcont2	1623	2053	245	245	430
s13207	2422	2399	118	106	142
s38417	8498	9220	358	359	377
Sum	**874k**	**924k**	**25k**	**23k**	**33k**

References

[1] G. D. Hachtel and F. Somenzi, *Logic Synthesis and Verification Algorithms*, Boston, MA, Kluwer Academic Publishers, 1996, 564 p.

[2] S. Hassoun and T. Sasao, *Logic Synthesis and Verification*, Boston, MA, Kluwer Academic Publishers, 2002, 454 p.

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

[4] M. Gao, Jie-Hong Jiang, Y. Jiang, Y. Li, S. Sinha, and R.K. Brayton, "MVSIS," in Notes of the IWLS, Tahoe City, June 2001.

[5] P. Bjesse and A. Boralv, "DAG-Aware Circuit Compression For Formal Verification," in Proc. of IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2004, pp. 42-49.

[6] A. Mishchenko and R. K. Brayton, "Scalable Logic Synthesis Using a Simple Circuit Structure," in Proc. of the 15th Intl. Workshop on Logic and Synthesis, Vail, Col., USA, June 7–9, 2006, pp. 15-22.

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

[8] 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.

[9] P. Fišer and J. Schmidt, "Small But Nasty Logic Synthesis Examples", Proceedings of the 8th. Int. Workshop on Boolean Problems, 2008, Freiberg, pp. 183–189.

[10] P. Fišer and J. Schmidt, "The Case for a Balanced Decomposition Process", in Proc. of the of 12th EUROMICRO Conference on Digital System Design, 2009, pp. 601–604.

[11] C. Yang and M. Ciesielski, "BDS: A BDD-Based Logic Optimization System", IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems Vol. 21, 2002, No. 7, pp. 866–876.

[12] L. Amarù, P.-E. Gaillardon, G. De Micheli, "Biconditional Binary Decision Diagrams: A Novel Canonical Representation Form", IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS), Vol. 4, No. 4, 2014, pp. 487-500.

[13] L. Amarù, P.-E. Gaillardon, G. De Micheli, "Boolean Logic Optimization in Majority-Inverter Graphs", Design Automation Conference (DAC), San Francisco, CA, USA, 2015.

[14] S. Yang, "Logic Synthesis and Optimization Benchmarks User Guide," Technical Report 1991−IWLS-UG-Saeyang, MCNC, Research Triangle Park, NC, January 1991, p. 45.

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

[16] 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), pp. 44-53.

[17] http://opencores.org