Improving FlexMap Tool to Explore Preprocessing and Post-processing Techniques

João Júnior da Silva Machado, Julio S. Domingues Junior
Leomar Soares da Rosa Jr., Felipe de Souza Marques

Group of Architecture and Integrated Circuits
Federal University of Pelotas

{jjdsmachado, jsdomingues, leomarjr, felipem}@inf.ufpel.edu.br

ABSTRACT

This paper addresses preprocessing and post-processing approaches applied through the logic synthesis flow. In this context, it proposes an extension to the FlexMap tool, allowing it to explore iterative methods and techniques during the technology mapping step. This extension is focused on handling EQN format and the obtained results demonstrate that such strategy is able to improve up to 50% when comparing to other mapping approaches.

Categories and Subject Descriptors

B.6.3 [**Logic Design**]: Design Aids – *automatic synthesis, optimization, simulation, verification.*

General Terms

Algorithms, Design, Experimentation, Performance, Verification.

Keywords

Technology Mapping, Logic Synthesis, EDA Tools.

1. INTRODUCTION

The Integrated Circuits (IC) manufacturing process has been improved along the last decades. However, even with advanced project techniques and Electronic Design Automation (EDA) tools, the IC design is not a trivial task. This is due to the short design cycle and the complexity of IC designs that usually have millions of logic gates. Thus, EDA tools are widely used by both the microelectronics industry and the academy.

Usually, the synthesis tools are mainly divided in three steps: high-level, logical and physical synthesis. The first one is responsible to synthetize a hardware description (Hardware Description Language - HDL) in a netlist of logic gates. The second step, the logic synthesis, performs technology independent optimizations and technology mapping. As stated in [7], it defines the main characteristics of the circuit, concerning area, power, delay, etc. Finally, the physical synthesis step takes the mapped netlist and basically performs place and route routines considering design rules of a manufacturing process, in order to generate the circuit layout [12].

This paper is mainly interested in the logic synthesis process. There are some algorithms that can be applied to optimize a logic description (technology independent optimizations), for instance, functional decomposition [2], factorization [4], two-level minimization [10], etc. Accordingly, to the standard cell methodology, the technology mapping procedure takes a logic description and transforms it into an interconnected netlist of cells. It defines the main structural characteristics of the circuit that will be reflected in the IC layout. Technology mapping may help to reduce the wire length leading to less congestion.

This way, the use of decomposition and factoring techniques associated to technology mapping steps can achieve better results in terms of area and power consumption. An example of this approach can be found in [8], which explores the factorization and decomposition to generate more optimized circuits. Kong [6] uses a similar approach, but exploring some SIS [11] scripts.

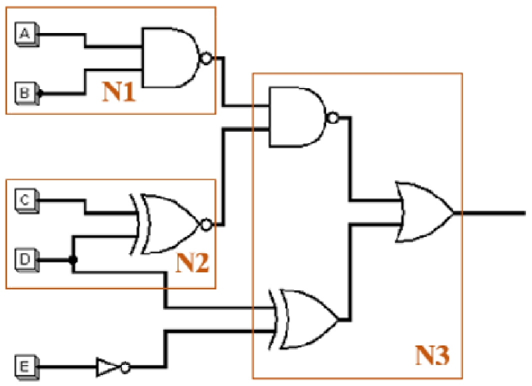
Since we are interested to work on logic synthesis algorithms, we have to handle different circuit descriptions, such as truth tables, sets of equations, graphs, etc. In this paper, we use both EQN and AIGER formats. The first one is developed by Synopsys, and it describes the logical function of a Boolean function through an equation. On the other hand, the AIGER format is provided by the Institute for Formal Models and Verification (FMV) from Austria. This format is used to describe an And-Inverter Graph (AIG), which is exclusively composed by 2-input AND nodes and edges that have an inversion attribute.

The main goal of this paper is to provide new features to the FlexMap tool [5]. This tool is a framework to develop new technology mapping methods. This way, it is possible to create alternative flows to build prototypes for different synthesis approaches. The main idea behind the FlexMap tool is the flexibility to customize the intermediate steps of the mapping procedure.

Currently, the FlexMap tool is only able to handle AIGER format files. However, in order to take advantage of some optimization algorithms implemented in other tools, we have made an extension to handle EQN files. This way, we can preprocess circuit descriptions using other tools to achieve technology independent optimizations. Therefore, the optimized description feeds the technology mapping procedures of the FlexMap tool. After the technology mapping, it can also write an EQN file that can be used in other commercial or academic tools, such as SIS and ABC [1].

The differences of EQN and AIGER formats are shown in Figure 1. While the first (Figure 1.b) represent the circuit of Figure 1.a using 15 nodes, the second (Figure 1.c) needs only 9 nodes to describe the same circuit. This is due to the fact that the AIGER format represents the circuit using the 2-input AND primitive on each of its nodes. Thus, large Boolean expressions have to be decomposed into 2-input gates. This does not happen when using the EQN format. In this case, a node can represent a more complex Boolean function, resulting in a netlist of cells.

The remaining of this paper is organized as follows: Section 2 addresses some important concepts to help the understanding of this paper. Our approach is introduced in section 3. Results are presented in section 4. Finally, we present some conclusions and future works.



(a)

```
aag 14 5 0 1 9
2
4
6
8
10
29
12 2 4
14 23 25
16 9 11
18 9 10
20 13 15
22 6 9
24 7 8
26 17 19
28 20 26
```

(b)

```
INORDER = pi_2 pi_4 pi_6 pi_8 pi_10;
OUTORDER = po_21;
po_21 = ![20];
N1 = pi_2 * pi_4;
N2 = (pi_6 * pi_8) + ((!pi_6) * (!pi_8));
N3 = (!(![N1] * ![N2])) + ((!pi_8) + pi_10);
```

(c)

Figure 1. Examples of AIGER (b) and EQN (c) descriptions from an IC design (a).

2. BACKGROUND

This section presents some concepts for a better understanding of this paper.

2.1 Technology Mapping

The technology mapping is a step applied into the logic synthesis process. This step consists on finding a set of interconnected logic cells to implement a given logic circuit, minimizing a given objective function. The technology mapping procedure can be divided into three steps: decomposition, matching and covering. The first stage aims to prepare the data structure for the mapping procedure. Thus, it usually generates a circuit representation using a specialized graph that often accepts a very small set of logic gates primitives. The matching procedure has to check equivalence between sub-graphs and cells of a library using Boolean/structural algorithms. This process will probably result in a set of matched sub-graphs where each sub-graph may overlap other sub-graph(s). In order to find a set of sub-graphs that minimizes an objective function, a covering algorithm is applied. It defines the set of cells that will be used to implement the circuit.

2.2 Factorization

Factorization is an operation that can be applied either in a single or in multiple Boolean expressions [4]. It tries to minimize the number of literals of Boolean expressions leading to area reduction [3]. Factorization algorithms are usually applied on two-level circuits that can be represented through sum of products (SOP). At the end of the process, a multilevel expression is achieved with a reduced number of literals. The factorization idea can also be applied on multiple Boolean expressions, aiming to find the best divisor expression that is common for more than one expression. It results in logic sharing which will lead to area reduction.

2.3 Functional Decomposition

The functional decomposition consists in to break a complex function into a smaller network in such a way that the original system's behavior keep up the same. This way, some constraints are satisfied and some objectives are optimized [2]. A given function F can be decomposed into two sub-functions G and H in such a way that the $G + H$ is Boolean equivalent to the original function F. Currently, this kind of methodology has been used to synthetize circuits that have to be mapped to fit a Field-Programmable Gate Array (FPGA).

3. PROPOSED EXTENSION

The FlexMap tool has focused on technology mapping. The main goal of this tool is to provide a flexible environment to setup a new mapping flow without using any extra programming. The tool has some methodologies to map circuits to a library using different techniques.

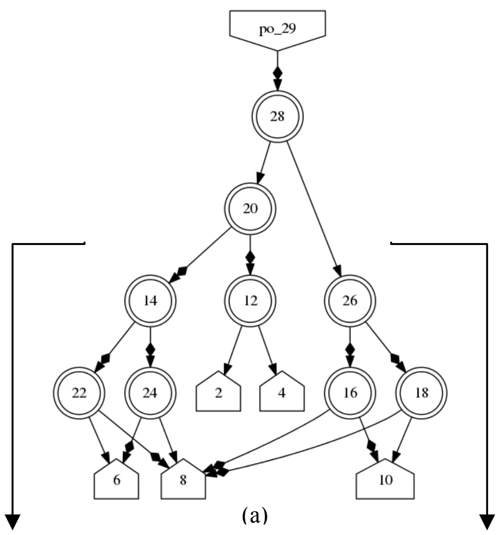
One of the main problems of this tool is related to the circuit representation. At the beginning, it was only taking AIGER files as input. As we have seen in Figure 1, this format may compromise the structural representation of the circuit because it is limited to inverters and 2-input AND gates representation. It means that any previously mapped circuit would have a completely different structure when represented with an AIG graph. In some cases, in order to be able to use some iterative mapping approach, it is necessary to preserve the netlist structure.

To fill this gap, we have proposed an extension to the tool. We are considering that it should have another possibility to read circuit netlists. One of the formats that are suitable to this is the EQN. Considering this possibility we can add new iterative features to the tool. Therefore, it can handle previously mapped circuits, even for other synthesis tools, and provide some iterative methodology to refine the solution. Furthermore, it is also able to write an EQN file to represent the structure achieved by the FlexMap tool.

The possibility to handle circuits described in EQN format allows to apply some steps of preprocessing and post-processing. Several approaches in the literature [8] [10] shows that a preprocessing step results in more optimized netlist circuits. In this sense, the FlexMap tool is able to make use of preprocessing methods of other synthesis tools, such as ABC and SIS.

When regarding post-processing steps, the FlexMap will be able to apply iterative methodologies in mapped circuits, using algorithms and techniques available in any other synthesis tool. Therefore, the proposed extension enables to refine the achieved solution until it meets the constraints of an objective function. Some approaches in the literature explore this idea, as the one described in [9].

All these optimizations are possible because the EQN format describes a logical function through Boolean equations. This way, it generates a description without change the circuit structure. Thus, it is able to achieve different decompositions for a given circuit, which will result in a better utilization of the available cells in a given library when composing the final circuit. Therefore, it is possible to deliver better mapping solutions. The following figure illustrates two possible descriptions using EQN format from a circuit described in AIGER format. Figure 2.a shows a structural representation of an AIG of the circuit illustrated in Figure 1.a. Figures 2.b and 2.c represent possible descriptions in EQN format from the AIGER format representation.



(a)

```
INORDER = pi_2 pi_4 pi_6 pi_8 pi_10;
OUTORDER = po_24;
[12]    = (!(pi_8)) * pi_10;
[14]    = pi_2 * pi_4;
[16]    = (!(pi_6)) * pi_8;
[18]    = pi_6 * (!(pi_10));
[20]    = ![12] * ![14];
[22]    = ![16] * ![18];
po_24 = ![20] + ![22];
```

(b)

```
INORDER = pi_2 pi_4 pi_6 pi_8 pi_10;
OUTORDER = po_20;
[12]    = !(pi_2 * pi_4);
[14]    = !((!pi_6 * pi_8) + (pi_6 * !pi_8));
[16]    = (!pi_8 * pi_10) + (pi_8 * !pi_10);
[18]    = !([14] * [12]);
po_20 = [16] + [18];
```

(c)

Figure 2. Possible descriptions using EQN format (b and c) to a circuit represented in AIG structure (a).

4. RESULTS

Some experiments were done in order to demonstrate that is possible to achieve different mapping with the use of the developed extension. The mapping strategy used in FlexMap tool for the logic-covering step is based on Simulated Anneling algorithm. The mapping method is focused on the optimization of circuit area using the cut-k algorithm [13] with k=3. This way, several decompositions methods were applied on the set of circuit from Microelectronics Center of North Carolina (MCNC) benchmark suite. In order to get mapped, the circuit descriptions were used as input to the FlexMap tool.

Table 1 shows the results in terms of number of elements (cells) necessary to map each circuit considering its initial decomposition. In order to compare the impact of preprocessing step, we performed the same SIS decomposition methods described in [6]. Similarly, to evaluate the initial decomposition quality, three different methods of decomposition from ABC tool were used. These methods are focused on decomposing the initial description aims to decrease the circuit area (*resyn* and *resyn2* command).

The *Circuit* column from Table 1 introduces the names of circuits from MCNC benchmark used on this experiment. The *AIGER* column shows the mapping result starting from the AIG description of the circuit. The *EQN* column refers to the mapping result using only decomposition based in logic gates primitives, such as AND-2, OR-2 and inverters. *SIS-1*, *SIS-2* and *SIS-3* columns refer to the decomposed circuits in SIS tool making use of the methods seen in [6]. *ABC-1*, *ABC-2* and *ABC-3* show the results obtained through decomposition methods present in ABC tool. Finally, the *Kong* column describes the results obtained by Kong in [6].

The results show that our proposed approach was able to explore different mapping solutions. The optimizations percentage on the different decompositions from the same mapping method reached up to 50% of improvement. Also, it is possible to verify, when comparing to the Kong approach, that decomposition methods from ABC tool deliver better results than decomposition methods from SIS tool if preprocessing and post-processing are available to use.

5. CONCLUSIONS AND FUTURE WORKS

This paper presented an extension to the FlexMap tool. This extension aims to increase the functionality, usability and compatibility of the FlexMap. With the use of this extension, it is possible to use circuits described in EQN format as inputs. This format allows representing the circuit without any changes to the original structure. This way, the new extension allows exploring different initial decomposition from an IC. In order to demonstrate the functionality of the proposed extension, several experiments were carried using different decompositions methods. The obtained results were better than the ones presented by Kong in [6], even using the same mapping methods available in SIS. Thus, the main goal of this experiments aims to demonstrate the possibility of explore different mapping solutions from an initial decompositions of the circuit. As future works, it is intend to develop new functionalities able to improve the mapping iteratively.

6. REFERENCES

[1] ABC - A System for Sequential Synthesis and Verification. Berkeley Logic Synthesis and Verification Group. [Online]. Available in http://www.eecs.berkeley.edu/~alanmi/abc/.

[2] Bertacco, V.; Damiani, M., "The disjunctive decomposition of logic functions," Computer-Aided Design, 1997. Digest of Technical Papers., 1997 IEEE/ACM International Conference on , vol., no., pp.78,82, 9-13 Nov. 1997.

[3] Brayton, R. K.; "Factoring logic functions," IBM Journal of Research and Development, vol.31, no.2, pp.187,198, March 1987.

[4] Brayton, R.K.; Rudell, R.; Sangiovanni-Vincentelli, A.; Wang, A.R., "MIS: A Multiple-Level Logic Optimization System," Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on , vol.6, no.6, pp.1062,1081, November 1987.

[5] FlexMap: A new Framework for Technology Mapping. Online. http://inf.ufpel.edu.br/gaci/?page_id=553.

[6] Kong, K; Shang, Y; Lu, R., "An Optimized Majority Logic Synthesis Methodology for Quantum-Dot Cellular

Automata," Nanotechnology, IEEE Transactions on , vol.9, no.2, pp.170,183, March 2010.

[7] F. S. Marques , L. S. da Rosa Jr. , R. P. Ribas , S. S. Sapatnekar , A. I. Reis, DAG based library-free technology mapping, Proceedings of the 17th ACM Great Lakes symposium on VLSI, March 11-13, 2007, Stresa-Lago Maggiore, Italy.

[8] Mishchenko, A.; Brayton, R.; Chatterjee, S., "Boolean factoring and decomposition of logic networks," Computer-Aided Design, 2008. ICCAD 2008. IEEE/ACM International Conference on , vol., no., pp.38,44, 10-13 Nov. 2008.

[9] Mishchenko, A.; Chatterjee, S.; Brayton, R.; Wang, X.; Kam, T., "Technology mapping with Boolean matching, supergates and choices". ERL Technical Report, EECS Dept., UC Berkeley, March 2005.

[10] Rudell, R. L.; Bartlett, K. A.; Brayton, R. K.; Hachtel, G. D.; Jacoby, R. M.; Morrison, C. R.; Sangiovanni-Vincentelli, A.; Wang, A., "Multi-level logic minimization using implicit don't cares," Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on , vol.7, no.6, pp.723,740, Jun 1988.

[11] SIS - Synthesis of both synchronous and asynchronous sequential circuits. Online. Available in: http://embedded.eecs.berkeley.edu/pubs/downloads/sis/.

[12] Ziesemer, A.; Lazzar, C., "Transistor level automatic layout generator for non-complementary CMOS cells," Very Large Scale Integration, 2007. VLSI - SoC 2007. IFIP International Conference on , vol., no., pp.116,121, 15-17 Oct. 2007.

[13] Cong, J.; Wu, C.; Ding, Y., “Cut ranking and pruning: Enabling a general and efficient FPGA mapping solution,” Proc. FPGA `99, pp.29-36.

Table 1. Evaluation considering area for the MCNC benchmarks ICs using different initial decompositions.

Circuit	AIGER	EQN	SIS-1	SIS-2	SIS-3	ABC-1	ABC-2	ABC-3	Kong
9symml	109	118	126	217	28	110	109	114	70
alu2	342	362	629	629	233	227	347	208	447
apex6	387	419	661	661	366	268	388	363	820
c8	139	144	259	259	66	47	138	51	140
cc	32	32	43	62	22	23	32	26	51
cht	170	180	307	307	79	59	171	82	129
cm150a	16	16	38	76	22	24	16	23	66
cm162a	25	24	34	52	19	15	24	18	55
cm163a	24	25	37	50	16	13	24	17	55
cu	34	34	43	66	27	25	34	25	61
example2	183	188	326	326	143	119	180	134	310
frg1	350	368	659	659	233	250	349	231	166
frg2	957	1113	1791	1791	446	426	963	361	715
i1	22	22	20	34	14	21	22	24	48
i2	150	153	232	232	121	142	150	144	209
k2	1313	1466	2289	2289	995	1350	1310	793	1520
lal	81	81	84	142	48	36	82	39	114
ldd	72	72	80	112	35	41	69	45	91
mux	67	70	84	123	32	61	67	23	58
pcle	24	24	31	62	24	16	24	26	79
pcler8	51	53	48	70	32	28	47	40	95
pm1	31	31	37	60	22	16	31	19	51
sct	87	89	95	152	42	89	89	31	86
term1	358	376	676	676	122	112	357	75	156
ttt2	280	294	508	508	97	85	281	78	197
unreg	48	48	59	112	50	28	48	63	117
vda	606	642	1020	1020	473	391	609	429	842
x1	838	915	1571	1571	364	841	841	167	359
x2	31	31	35	57	19	17	33	22	112
Total	**6827**	**7390**	**11822**	**12375**	**4190**	**4880**	**6835**	**3671**	**7486**
Ratio	**1,097**	**1,013**	**0,633**	**0,605**	**1,787**	**1,534**	**1,095**	**2,039**	
Optimized	**8,803**	**1,282**	**-57,921**	**-65,309**	**44,029**	**34,812**	**8,696**	**50,962**	