DAG Based Library-Free Technology Mapping

F. S. Marques[1], L. S. Rosa Jr[1], R. P. Ribas[1], S. S. Sapatnekar[2], A. I. Reis[1,3]

[1]Instituto de Informática – UFRGS
Porto Alegre, RS, Brazil
{felipem, leomarjr, rpribas}@inf.ufrgs.br

[2]EECS - University of Minnesota
Minneapolis, MN, USA
sachin@ece.umn.edu

[3]Nangate Inc.
Menlo Park, CA, USA
are@nangate.com

ABSTRACT

This paper proposes a library-free technology mapping algorithm to reduce delay in combinational circuits. The algorithm reduces the overall number of series transistors through the longest path, considering that each cell network has to obey to a maximum admitted chain. The number of series transistors is computed in a Boolean way, reducing the structural bias. The mapping algorithm is performed on a Directed Acyclic Graph (DAG) description of the circuit. Preliminary results for delay were obtained through SPICE simulations. When compared to the SIS technology mapping, the proposed method shows significant delay reductions, considering circuits mapped with different libraries.

Categories and Subject Descriptors

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

General Terms

Algorithms, Performance, Design, Experimentation, Theory.

Keywords

Logic Synthesis, Switching Theory, Technology Mapping, Library Free Synthesis, Virtual Libraries.

1. INTRODUCTION

Technology Mapping (TM) is the step of logic synthesis that chooses the cells that will be used to implement a circuit in a given technology. Normally, the cells are chosen from a pre-characterized library [1-4]. First methods for Technology Mapping used trees as the initial description of the circuit to be mapped. More recent methods are based on Directed Acyclic Graph (DAG) representations that allow duplicating logic to some extent to increase speed. Another important contribution to technology mapping was Boolean matching [5], where the matching of a portion of the circuit and a cell from the library is done by comparing the Boolean function of the candidates, instead of the structure. Structural comparison would not be able to find all matches.

GLSVLSI'07, March 11–13, 2007, Stresa-Lago Maggiore, Italy.

In the early phase of technology mapping, it was considered that the use of a cell generator [6] would enable the use of larger virtual (built on demand) cell libraries. Unfortunately, the use of such approaches was not widely verified in a commercial level, even if other references suggest that the increased number of cells in a library could lead to significant improvements in the quality of the final design [7-10]. A recent approach presented in [11] suggests that the addition of some custom cells to a library can improve the speed of the final circuit. Recently, some methods for generating efficient cell networks were proposed [12-15], including a method [15] to compute the minimum number of transistors in series needed to implement an arbitrary Boolean function. These improvements were presented only at the cell level, lacking of an efficient method for mapping a larger circuit.

The contribution of this paper is to combine the method for Boolean computation of the number of series transistors presented in [15] with a state of the art technology mapping algorithm inspired by the approach presented in [4]. Significant gains are obtained in delay due to both aspects combined into the proposed mapping tool. The algorithm is library-free, as it chooses the transistor configuration for the cells that will have to be created through a cell generation tools in a subsequent step.

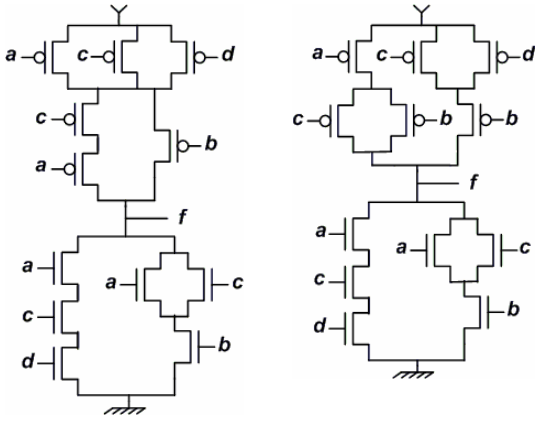
This paper is organized as follows. Section 2 presents the background. It describes why the method used to compute series transistors is Boolean, and presents the rationale that relates series transistors to circuit delay. Section 3 presents the proposed algorithm. Results are presented in section 4 and conclusions are presented in section 5.

2. BACKGROUND

The method used in this paper to compute the number of series transistor in a cell network is a Boolean method. It computes the lower bounds [15] for the number of series transistors in the longest pull-up and pull-down chains of a cell implementing a given logic function. Boolean methods are able to overcome the structural bias [18] of the circuit being mapped, because they do not depend on the DAG structure, but only on the function being mapped. Another important point is that the associative methods to compute series transistor constraints used in [6, 7, 8, 9, 10] are monotonically increasing with the association, meaning the association of two functions will always have more transistors in series. The Boolean method is non monotonic, meaning the association of two functions can reduce the number of transistors in series. The differences between the Boolean method (lower bound) used in this paper and the well known complementary series-parallel approach (CSP) are highlighted in Table 1.

Table 1. Methods for computing length of transistor chains.

	Logic used	
Characteristic	CSP	Lower Bound
Cells used	Restricted to series-parallel complementary	Not always topologically complementary
Constraint computation	Monotonically increasing and structural	Non-monotonic and Boolean
Minimum length transistor chains	Not guaranteed	Guaranteed
Structural bias	Very dependent on initial description	Reduced by Boolean computation of constraints
Approaches	[6, 7, 8, 9, 10]	[15]



a) CSP CMOS cell b) Lower bound cell

Figure 1. CSP and NCSP topologies for transistor networks

For computing the lower bound we use a modified version of ESPRESSO-SIGNATURE [19] that does not generate the final SOP, only computes the length of minimum transistor chains. Our algorithm also uses a method to produce cell networks with minimum length transistor chains for the selected functions. Fig. 1 shows a comparison between a CSP and a lower bound implementation for the same function. Notice that the number of series transistors in the pull-up is reduced in Fig.1.b, leading to a faster implementation. More details can be found in [15].

3. LLWF ALGORITHM

The Library-Less Wavefront (LLWF) technology mapping algorithm is outlined in Fig. 2. It uses matching and covering routines presented in Fig. 3and Fig 4, respectively. Our algorithm uses a data structure very similar to the AIG in OAgear [20]. The library-less wavefront algorithm is illustrated by the Fig. 5, considering the initial circuit of the Fig. 5.a, which is a 2-input decomposition of the Boolean function f presented in section 2 (Fig. 1). Fig. 5.d shows the possible heads and all the discovered matches. The labeled vertical dashed lines represent the successive *heads* of the wavefront. In the algorithm proposed here, the matching generation window is given by the *wave_width*. Hence, the pattern matches are generated in the interval *[head - wave_width : head]*, as described by the algorithm in Fig. 3. The covering algorithm relies on a function for computing the lower bound of series transistors. As this procedure needs to be fast, we developed a routine similar to ESPRESSO-SIGNATURE [19]. The covering algorithm is shown in Fig. 4, and it minimizes the accumulated sum of transistor lengths along the critical path. This initial cost function is a first order approach, but our results show that it correlates well with circuit delay. A possible explanation is that the reduction of the number of series transistors decreases the logical effort [16] of the cells. In the future we intend to use a more precise delay model during the covering approach. For the moment our first order approach produces results that demonstrate the contribution of the method.

```
Procedure LLWF_mapper(max_pu, max_pd, wave_width) {
   remove_all_inverters();
   levelize_circuit();
   highest_level = highest level of the circuit;
   head_level = 1;
   while (head_level ≤ highest_level) {
      head_nets = list of all nets on head_level;
      foreach net, n in head_nets {
         /*Generate all matches considering a set of
         constraints*/
         generate_matches(n, max_pu, max_pd, head,
wave_width);
         /*Select the best match for the net n*/
         covering_algorithm(n);}
      increment head_level;}
   add_inverters();
}
```

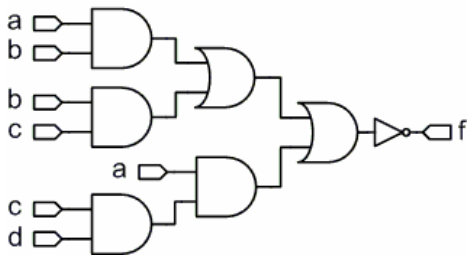
Figure 2. Main Algorithm.

```
Procedure generate_matches(n, max_pu, max_pd, head,
wave_width) {
   - In the DAG, generate all the pattern matches for the net
     n, such that the search for pattern matches is performed
     in the interval [head – wave_width : head]; At this point,
     other constraints can be used to limit the match
     generation;
   foreach pattern match, pat in the list of pattern matches {
      /* Compute lower bound for pull-up and pull-down
       planes*/
      cost_pu = compute_lower_bound_pu(pat);
      cost_pd = compute_lower_bound_pd(pat);
      if (cost_pu <= max_pu && cost_pd <= max_pd) {
         - Store pat as a logic_cell;
         - Make logic_cell, a driver of n in the DAG
         representation;
         - Connect all inputs of logic_cell to their
         correspondent nets;
      }
   }
}
```

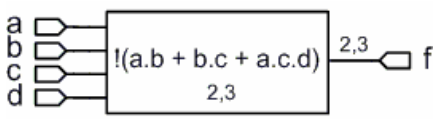
Figure 3. Matching Algorithm.

```
Procedure covering_algorithm(n) {
   - Compute the sum of pull-up for all cells driving net n;
   - Compute the sum of pull-down for all cells driving net n;
   - Select the cell with the lowest sum of pull-up and pull-
   down;
   - Disconnect all driver cells on n, except the selected cell,
   and perform a cleanup operation on their exclusive inputs;
}
```

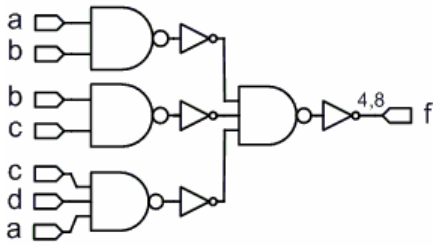
Figure 4. Covering Algorithm.



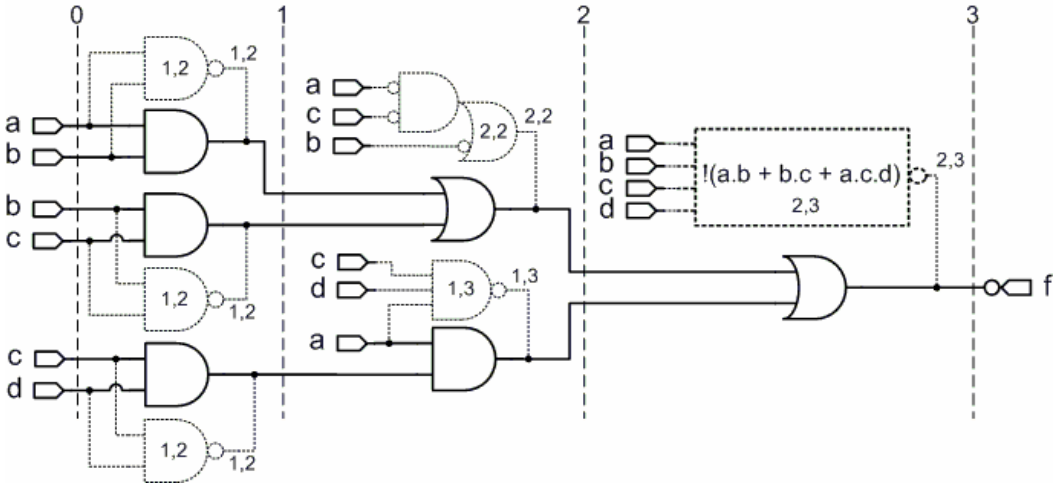
a) 2-Input decomposition



b) circuit mapped using *wave_width = 3*



c) circuit mapped using *wave_width = 2*



d) best matches for each net in the circuit

Figure 5. The LLWF technology mapping method

In the algorithm, two constraints are used to validate the patterns: *max_pu* and *max_pd*. Both limit the maximum number of series transistors for the pull-up (*max_pu*) and pull-down (*max_pd*) chains for each match of a given circuit net. As an example of the application of the algorithm, assume the following constraint values: *max_pu = 2*, *max_pd = 3* and *wave_width = 3*. Initially, the *head* starts at level 0 (primary input nets). It advances level-by-level and the match generation is done for all nets on the *head* level. After the matching generation for a given net *n*, the covering algorithm is immediately invoked to choose the best match for *n*. These steps will be repeated until the *head* reaches the highest level in the circuit. In the Fig. 5.d, when the *head* reaches the level 1, the 2-Input NAND gates are added as drivers of their respective nets in the circuit, creating multi-source nets. After the covering algorithm selects the best match for the last net on level 1, the *head* is moved to level 2. As the *wave_width* is equal to the highest level of the circuit, on level 2 and 3, matches are generated until the primary inputs. Finally, considering the inversion flags in the circuit representation, inverters are inserted when it is necessary. Fig. 5.b shows the mapped circuit under initial constraints. If the *wave_width* is reduced to 2, the circuit in the Fig. 5.c is obtained. This shows how the *wave_width* affects the quality of the mapped circuit.

All cells in the final circuit are enforced to have an equal or smaller pull-up compared to pull-down. This is achieved by changing the polarity of cell inputs and outputs and exchanging pull-up and pull-down networks. Besides the constraints *max_pu* and *max_pd*, another set of restrictions can be used to avoid an excessive number of matches. For instance, the number of variables and/or the number of literals can also limit pattern matches. The matches are not limited to fanout-free (tree) regions; i.e. the match generation search process performs its search across fanout, since these nets are in the interval *[head - wave_width : head]*. However, it can be easily limited to fanout-free regions testing the fanout of each net in the search space. This technique can be used in order to save area.

4. RESULTS

In this section we present results of the algorithm developed in this paper. It was implemented in a tool called VIRMA-WF, which has been written in Java. All results were generated on a PC workstation running Windows XP using an AMD Athlon

64/3200+ processor. First experiment investigates the effect of the *wave_width* parameter. Second experiment compares the technology mapping performed by VIRMA-WF, using SIS [17] as a reference, for a set of benchmark circuits.

First study to analyze the effect of varying the *wave_width* sizes. The effect on delay is illustrated in Fig. 6 using four benchmark circuits and varying the *wave_width* from 1 to 6. As it happens for the original wavefront [4], for widths of more than 4 the delay is constant or presents slight improvements. Since the width 4 is more practical, this wavefront size was chosen in order to realize the comparison between SIS and LLWF technology mapping algorithms.

The second experiment is a comparison between SIS technology mapping and our method. The circuits were first decomposed into inverters and 2-Input NAND/NOR gates using SIS. Next, we performed technology mapping, using SIS and our method, for all benchmark circuits. Finally, using our cell generator, NCSP and CSP CMOS transistor networks are derived for the mapped circuits produced by our method and by SIS, respectively. Results are shown in five different tables. In tables 2-6, the first columns show the name of the circuit. The labels of the following columns describe the cell libraries used during the technology mapping. The columns *33-4* and *lib2* show results for the cell libraries *33-4.genlib* and *lib2.genlib*, respectively, mapped by SIS targeting minimum delay. Since the cell libraries *33-4.genlib* and *lib2.genlib* have cells with costs equal or lesser than *3* for the pull-up and pull-down planes, and few cells where PU or PD cost can be *4* (such as cells found in the *lib2.genlib*), all circuits were mapped by our method using *wave_width = 4* and the constraints *3,3* and *3,4* to limit PU and PD costs. The label T (e.g. *(3,3)-T* and *(3,4)-T*) indicates that the mapping was limited to trees (fanout free regions). The label D (e.g. *(3,3)-D* and *(3,4)-D*) indicates that the mapping was allowed to duplicate logic, resulting on DAG mapping.

Table 2 shows the accumulated sum of series transistors on the pull-up and pull-down planes of each cell on the longest path of the circuit. It is noticeable that VIRMA-WF reduces the accumulated transistor chains along the longest path.

In order to prove that the reduction of transistor in the pull-up and pull-down planes can reduce the circuit delay, we used SPICE simulation to estimate delay. The transistors used on the SPICE description have fixed size. Table 3 presents a delay comparison between the SIS technology mapping VIRMA-WF technology mapping. The second column (33-4 (ns)) shows delay values expressed in nanoseconds for circuits mapped by SIS. The columns 3-7 show normalized values correspondent to the delay values of the second column. Our method provides better results than SIS results, with average delay reductions of about 27% and 33% considering virtual cell libraries restricted by the constraints *3,3* and *3,4*, respectively. The technology mapping limited to tree (T) regions performed by our method also shows improvements of 13%-15% in average.

Area comparison, considering the number of transistors of each circuit, can be seen in table 4. Due to logic duplications during the technology mapping, inherent to DAG mapping, our method can increase the area. The area penalty for using our technology mapping algorithm is 18% and 31% in average for the virtual cell libraries *3,3* and *3,4*, respectively. There are cases where the average area increase is negligible. It happens for the technology mapping limited by fanout. Although for these cases the delay gains were not maximized, a good area/delay trade-off is still achieved.

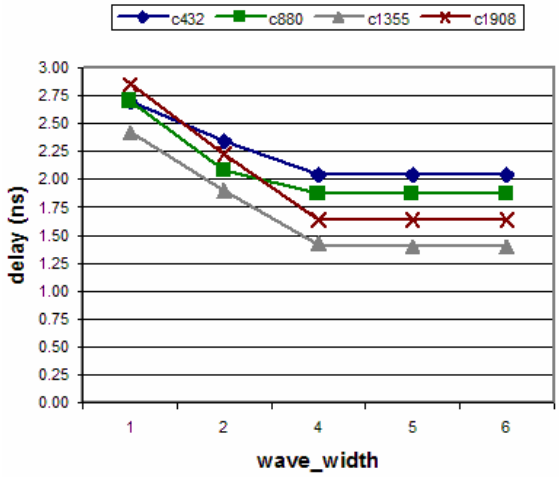


Figure 6. The effect of varying the *wave_width*

Table 5 shows the execution times for SIS and VIRMA-WF, given in seconds. The VIRMA-WF is more time consuming than SIS. As the time values show, they are not proportional to the size of the circuit. For instance, considering the virtual cell library *(3,4)*, the circuit c499 uses more time than the circuit c3540. However, c499 is smaller than c3540. This is mainly due to the complexity of the lower bound calculus for each match, and also to the number of generated matches during the technology mapping process.

Table 2. Accumulated pull-up and pull-down along the longest path of the circuit.

	SIS				VIRMA-WF							
	33-4		lib2		(3,3)-T		(3,3)-D		(3,4)-T		(3,4)-D	
Bench.	**SPU**	**SPD**	**SPU**	**SPD**	**SPU**	**SPD**	**SPU**	**SPD**	**SPU**	**SPD**	**SPU**	**SPD**
c1355	31	52	31	55	34	45	25	30	31	46	19	27
c1908	41	58	40	56	39	52	30	38	37	50	27	40
c3540	60	78	54	69	42	67	39	60	39	69	33	52
c432	39	55	34	52	33	49	31	43	25	47	24	41
c499	29	35	24	35	27	32	25	27	24	33	18	26
c6288	124	243	131	247	153	213	95	113	153	213	88	125
c880	32	41	21	39	26	35	21	32	20	36	16	38

Table 3. Delay for VIRMA-WF vs. SIS.

	SIS		VIRMA-WF			
Bench.	**33-4 (ns)**	**lib2**	**(3,3)-T**	**(3,3)-D**	**(3,4)-T**	**(3,4)-D**
c1355	2.32	1.07	0.94	0.61	0.93	0.52
c1908	2.53	1.08	0.87	0.65	0.87	0.71
c3540	3.37	1.04	0.94	0.90	0.91	0.65
c432	2.70	0.88	0.82	0.76	0.79	0.70
c499	1.87	0.91	0.82	0.71	0.80	0.64
c880	2.27	0.91	0.83	0.77	0.82	0.80
Average		**0.98**	**0.87**	**0.73**	**0.85**	**0.67**

Table 4. Area for VIRMA-WF vs. SIS.

	SIS		VIRMA-WF			
Bench.	**33-4**	**lib2**	**(3,3)-T**	**(3,3)-D**	**(3,4)-T**	**(3,4)-D**
c1355	2140	1.03	0.92	0.87	0.89	1.15
c1908	2390	1.01	1.02	1.19	0.94	1.11
c3540	4410	1.09	0.99	1.32	1.05	1.38
c432	790	1.13	1.13	1.37	1.08	1.39
c499	1484	1.00	1.04	1.00	1.00	1.26
c880	1256	1.00	1.08	1.33	1.03	1.54
Average		**1.05**	**1.03**	**1.18**	**1.00**	**1.31**

Table 5. CPU time for VIRMA-WF vs. SIS.

	SIS		VIRMA-WF			
Bench.	**33-4**	**lib2**	**(3,3)-T**	**(3,3)-D**	**(3,4)-T**	**(3,4)-D**
c1355	0.4	0.3	13.0	494.1	27.8	542.4
c1908	0.6	0.5	15.3	404.7	19.1	766.3
c3540	2.9	2.6	41.3	469.2	44.6	777.7
c432	0.2	0.1	5.5	25.6	6.3	36.0
c499	0.3	0.3	12.7	992.8	27.3	1641.3
c880	0.3	0.2	1.7	93.0	2.1	281.2
Total	**4.7**	**4.0**	**89.5**	**2479.3**	**127.1**	**4044.9**

Table 6. C6288 results.

	SIS		VIRMA-WF			
	33-4	**lib2**	**(3,3)-T**	**(3,3)-D**	**(3,4)-T**	**(3,4)-D**
Delay	10.20	1.02	1.00	0.57	1.00	0.63
Area	9444	1.01	1.00	1.90	1.00	2.07
CPU time	9.0	8.6	7.6	1028.4	8.0	1304.0

Table 6 shows results for the benchmark circuit c6288. For delay and area results, the column '33-4' presents, respectively, absolute values in nanoseconds and number of transistors. The following columns show the correspondent relative values. The CPU time is expressed in seconds for all libraries. Delay gains are very significant for the libraries *(3,3)* and *(3,4)*, when DAG mapping is applied. However, the area penalty is high. The c6288 is a multiplier composed by regular logic blocks, and it has several regions that are not fanout-free. Therefore, best matches that cross fanout, will probably be best matches for other regions, resulting in many duplications. This area penalty can be reduced by allowing duplication of logic only for timing critical regions.

The prototype implemented to obtain the experimental results is devoted to prove our concepts. Our results show considerable delay gains. Nevertheless, area results show that we have to look for better area/delay trade-offs. We expect to find it by allowing duplication only in critical regions of the circuit. It can also decrease the CPU time, since the number of matches will be reduced. Another possibility to reduce CPU time is to store pre-computed lower bounds in a hash table, to avoid repeated computations.

5. CONCLUSIONS

We have presented a library-less technology mapping algorithm to reduce delay in combinational circuits. A comparison among the tradition technology mapping using SIS and our method using different virtual cell libraries shows delay reductions from 6% to 48%. For some circuits, better delay means high penalty in area. The VIRMA-WF technology mapping limited to fanout-free regions produce circuits with negligible area increase and with delay improvements around 15% in average. In order to find a good trade-off between area and delay, the VIRMA-WF algorithm can be extended using a mix of tree-mapping on non timing critical regions and DAG-mapping on timing critical regions of the circuit, as suggested in the previous section.

The method presented here can be implemented in a non-disruptive way in existing design flows, if a cell/library generation tool is available. After mapping, the bespoke logic functions must be generated as cells to compose a library that is used for place & route and design closure of the mapped logic network. Future works will address this issue.

6. ACKNOWLEDGMENTS

This research was partially supported by CNPq/PNM and CAPES Brazilian Funding Agencies.

7. REFERENCES

[1] K. Keutzer, "Dagon: Technology binding and local optimization by DAG matching", In Proc of the 24th Design Automation Conference, June 1987, pp. 341-347.

[2] E. Lehman, Y. Watanabe, J. Grodstein, and H. Harkness, "Logic decomposition during technology", *IEEE Transactions on Computer-Aided Design*, August 1997, 16(8):813-834.

[3] Y. Kukimoto, R.K. Brayton, and P. Sawkar, "Delay-optimal technology mapping by DAG covering", In Proc of the DAC'98, 1998. pp. 348-351.

[4] L. Stok, M.A. Iyer, and A.J. Sullivan, "Wavefront Technology Mapping", In Proc. of the DATE'99, Germany, 1999.

[5] F. Mailhot, and G. DeMicheli. "Algorithms for technology mapping based on binary decision diagrams and on Boolean operations", *IEEE Transactions on CAD for IC and Systems*, vol. 12 n° 5, May 1993, pp. 599-620.

[6] M. Berkelaar, and J. Jess, "Technology mapping for standard-cell generators", In Proc. Int. Conf. Computer-Aided Design, Santa Clara, CA, Nov. 1988, pp. 470-473.

[7] K. Keutzer, K. Kolwicz, and M. Lega, "Impact of library size on the quality of automated synthesis", ICCAD 1987, pp. 120-123.

[8] K. Scott, and K. Keutzer, "Improving cell libraries for synthesis", In Proc. of CICC, 1994, pp. 128-131.

[9] C. Sechen, and B. Guan, "Large standard cell libraries and their impact on layout area and circuit performance", In Proc. of ICCD, 1996, pp. 378-383.

[10] S. Gavrilov, A. Glebov, S. Pullela, S. Moore, A. Dharchoudhury, R. Panda, G. Vijayan, and D. Blaauw, "Library-less synthesis for static CMOS combinational logic circuits", in Proc. of ICCAD, 1997, pp:658 – 662.

[11] R. Roy, D. Bhattacharya, and V. Boppana, "Transistor-level optimization of digital designs with flex cells", *IEEE Computer*, Feb. 2005, pp. 53-61.

[12] M. Kanecko and J. Tian, “Concurrent cell generation and mapping for CMOS logic circuits”, In Proc. of ASPDAC97, 1997, pp. 247–252.

[13] R. Poli, F. Schneider, R. Ribas, and A. Reis. “Unified theory to build cell-level transistor networks from BDDs”. In Proc. of SBCCI 2003, pp.199 – 204.

[14] K. Tanaka, and Y. Kambayashi,. “Transduction method for design of logic cell structure”, In Proc. of ASPDAC2004, pp. 600–603.

[15] F. Schneider, R. Ribas, S. Sapatnekar, and A. Reis, “Exact lower bound for the number of switches in series to implement a combinational logic cell”, In Proc of ICCD05, 2005, pp. 357-362.

[16] I. Sutherland, B. Sproull, and D. Harris “Logical Effort: Designing Fast CMOS Circuits”, *Morgan Kaufmann*, 1999.

[17] E.M. Sentovich, K.J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P.R. Stephan, R.K. Brayton, A. Sangiovanni-Vincentelli, “SIS: A system for sequential circuit synthesis”, Technical Report No. UCB/ERL M92/41, EECS Department, University of California, Berkeley, 1992.

[18] S.Chatterjee, A.Mishchenko, R.Brayton, X.Wang, T.Kam. “Reducing Structural Bias in Technology Mapping”. IEEE TCAD, accepted for future publication, 2006.

[19] P.McGeer, J.Sanghavi, R.Brayton, A.Sangiovanni-Vicentelli. “ESPRESSO-SIGNATURE: a new exact minimizer for logic functions”. IEEE Transactions on VLSI, Volume 1, Issue 4, Dec. 1993 Pp:432 – 440.

[20] Z.Xiu, D.A.Papa, P.Chong, C.Albrecht, A.Kuehlmann, R.A.Rutenbar, I.L.Markov. “Early research experience with OpenAccess gear: an open source development environment for physical design”, In Proc. of ISPD05, pp. 94-100.