A TREE-BASED TRANSISTOR GATE MATCHING SOLUTION FOR EFFICIENT LAYOUT IMPLEMENTATION

[1]Diogo C. da Silva, [1]Fábio R. Pereira, [1]Leomar S. da Rosa Jr., [2]André I. Reis, [1]Renato P. Ribas

[1]Instituto de Informática – UFRGS, Porto Alegre, Brazil
[2]Nangate Inc., Sunnyvale, CA, USA
{dcsilva, frpereira, leomarjr, rpribas}@inf.ufrgs.br, are@nangate.com

ABSTRACT

Circuit layout generation is one of the last steps in the physical synthesis workflow. In this step, the circuit logic description is translated into a physical representation in terms of planar geometric shapes corresponding to the patterns of metal, oxide, or semiconductor layers that make up the components of the final integrated circuit. As a result, the final circuit area is closely related to the quality of the generated layout. Matching transistor gates is an essential task for achieving an efficient layout implementation. To deal with this issue, this paper proposes a tree-based algorithm as a solution for (i) finding Eulerian paths inside a transistor network description and (ii) reaching the maximal gate matching between the pull-up and pull-down logic planes.

1. INTRODUCTION

Nowadays, one of the most established design flows for digital microelectronics design is the standard-cell approach. This flow consists in using pre-built logic cells to automatically compose the digital circuit. Synthesis tool maps the target circuit according to a given logic cell library, based on costs concerning area, signal propagation delay and power consumption [1,2]. Usually, cell libraries are composed of a few tens of logic cells, due to the engineering effort to include a new one. These logic cells have been previously characterized and tested, and all information about their behavior is described in a database which, in turn, is used during the technology mapping procedure.

Some researchers have observed that large cell libraries could lead to a better circuit implementation [3]. However, the number of potential logic function increases exponentially with the number of inputs. Therefore, it is not possible to characterize and implement every logic function possible in order to compose a given library. The processes of characterization and layout generation are extremely computing demand, making the possibility of having large cell libraries unfeasible [4].

An alternative to technology mapping is the concept of library-free circuit design. In this approach, the technology mapping tool does not use a cell library when performing the mapping procedure. All logic cells needed to compose the circuit are delivered by an automated cell generator. The main disadvantage of this approach is the lack of cell characterization data. All of the mapping procedure is performed employing non-characterized information. Generally, transistor count and transistor stack are used as cost by library-free technology mapping tools [1,5,6]. Consequently, more realistic information regarding layout area cannot be obtained.

When regarding layout, it is important to utilize small logic cells to guarantee better implementations of digital circuits. In order to achieve such a goal, it is desirable that the transistors composing the logic planes of a given logic cell can be aligned. Such a situation would eliminate the need for unnecessary internal connections between the transistor gates, possibly minimizing cell dimensions.

The purpose of this work is to present a solution to achieve networks with maximal matching between transistor gates at a symbolic layout (topological level). The main motivation is having, in the future, an automated workflow capable of being used by digital circuit designers who wish to obtain area results for different logic cell implementations [7,8,9] without having to generate their layout. Furthermore, this technique could be useful in a library-free approach, which would use the estimated layout area as a more accurate cost to perform circuit mapping.

The remainder of this paper is organized as follows. Section 2 describes the algorithm for searching Eulerian paths. Section 3 offers a solution for finding gate matches between logic planes. Some results are given in Section 4. Finally, Section 5 presents conclusions and future works.

2. SEARCHING EULERIAN PATHS

In graph theory, Eulerian paths are paths that visit each edge in a graph exactly once. They were first discussed by Leonhard Euler while solving the famous problem of the Seven Bridges of Königsberg in 1736 [10,11]. Graphs containing such paths are called traversable. A graph is traversable when it contains zero or two vertices of odd degree [12]. Fleury's algorithm [13] is widely used for searching Eulerian paths in traversable graphs. In short, the algorithm involves starting from one of the two odd vertices and traversing the graph, crossing all edges only once and finally arriving at the other odd vertex. If there are no odd vertices, any vertex can be used as a starting point. In modern microelectronics, this concept is very important, since a network of transistors can be represented as a

multigraph where Eulerian paths may be used to define the positioning of transistors in a layout implementation.

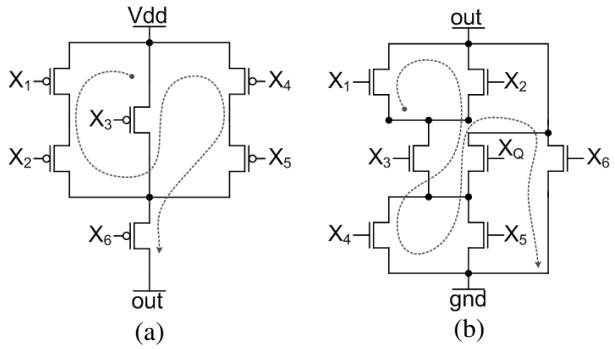


Figure 1 – (a) PMOS transistor network and (b) NMOS transistor network showing possible Eulerian paths

When a graph contains more than two vertices of odd degree, dummy edges may be inserted between them, making their degrees even. If enough dummy edges are inserted, any connected graph can be made traversable. Fig. 1b illustrates the insertion of a dummy transistor (X_Q) in a NMOS transistor network containing four nodes of odd degree. Note that the dummy transistor can be inserted between any pair of odd nodes.

Given a transistor network containing 'n' nodes of odd degree ($n > 0$), the number of dummy transistors required to make it traversable (d) can be obtained from the equation $d = (n - 2)/2$. For the example illustrated in Fig. 1b, only one dummy edge is necessary.

A given logic plane in a disjoint transistor network may contain several Eulerian paths. In order to find all possible paths, the following steps are applied:

1. Each logic plane in the transistor network is converted into a multigraph representation.
2. The number of dummy edges to be inserted in the graph is determined using the equation described above.
3. Dummy edges are inserted between all possible pairs of odd vertices.
4. The multigraph is then traversed, starting at each of the odd vertices (or all the vertices, if there are none). The number of dummy edges used in a path is limited to the amount obtained in step 2.
5. All Eulerian paths found are stored in a tree-like structure to be analyzed by the gate matching algorithm. The tree nodes represent the gates of transistors in the network, and paths between the root and the leaves represent Eulerian paths. Fig. 2 shows partial path trees for each of the logic planes illustrated in Fig. 1.

3. GATE MATCHING

The gate matching process consists in finding a pair of Eulerian paths – one for the NMOS plane and one for the PMOS plane of the same transistor network – containing the same sequence of transistor controlling signals. It is important because aligning gates reduces the complexity of internal connections between the NMOS and PMOS planes. In this context, a good match could benefit the routing procedure, which is one of the most critical steps when generating a cell layout implementation. In addition, the layout area requirements could be minimized, since there is no need for extra rows to connect the crossing polysilicon gates. This leads to a smaller layout implementation, and avoids the use of an extra layer of metal in order to connect unaligned gates. Fig. 3 illustrates two possible symbolic layout solutions (aligned and unaligned) for the cell shown in Fig. 1.

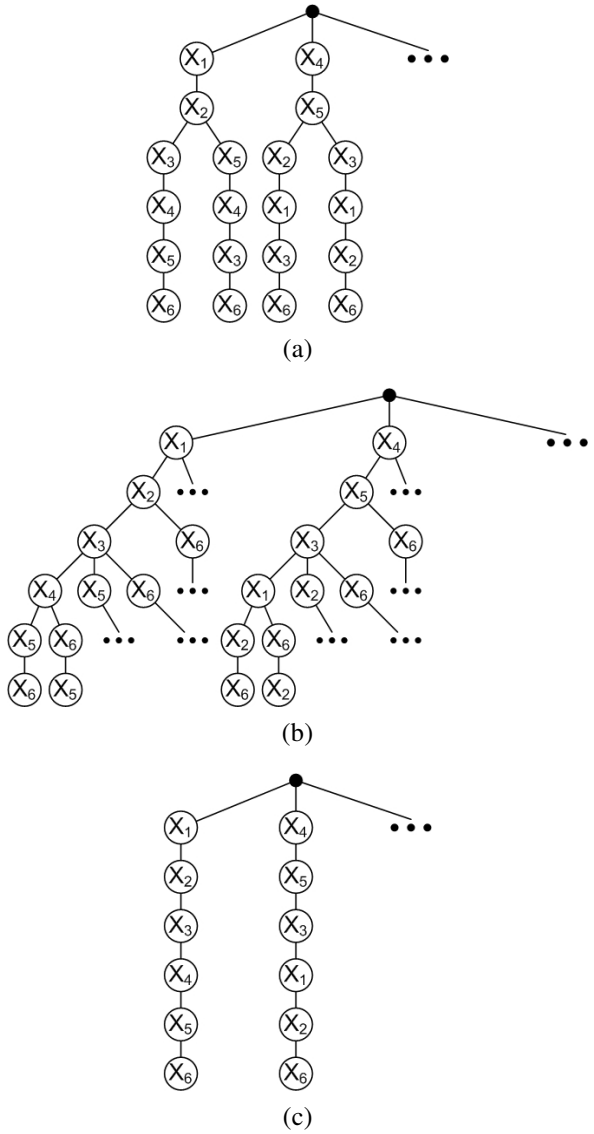


Figure 2 – Partial tree for the cell in Fig. 1, before (a, b) and after (c) the gate matching algorithm

To achieve gate matching, the following algorithm is proposed:

- Two trees obtained as described in Section 2 are simultaneously traversed in a recursive manner, starting at their roots.

- Each node in a tree is compared to its counterpart in the other tree. If a given node does not exist in one of the trees, it is removed, along with its child nodes.
- At the end of the algorithm, only corresponding nodes remain. These nodes represent matching gates in a pair of Eulerian paths.

Fig. 2c illustrates the partial tree for the cell described in Fig. 1 after the gate matching algorithm has concluded. Note that only one tree is shown, since the two resulting trees are identical.

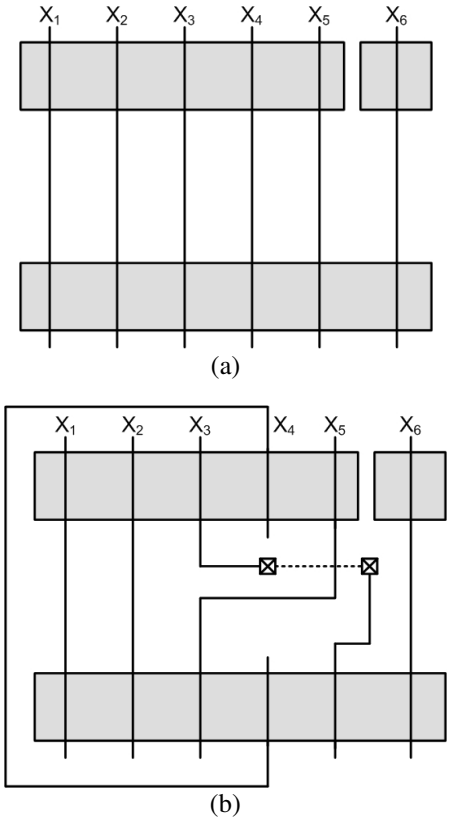


Figure 3 – Two possible symbolic layouts for the cell in Fig. 1, showing matched (a) and mismatched gates (b).

4. RESULTS

The algorithms proposed in this paper were implemented in the Java programming language. For the purpose of demonstrating the relevance of matching gates, a hypothetical 4-input logic function was evaluated. The function was implemented in the NCSP logic style [7], resulting in a logic cell comprised of twenty transistors (ten transistors in each plane). The algorithms were then applied to the cell, resulting in a new version with aligned gates. A commercial tool [14] was used to generate symbolic layouts and perform the routing step for both versions of the cell. Fig. 4 shows the outputs generated by the tool, illustrating the impact of using the techniques presented here.

As can be seen, gate matching simplifies the routing of internal connections in the logic cell. Shorter metal and polysilicon lines are used in the aligned solution, since the gates in opposite planes are physically closer to each other. Although it was not analyzed in this work, signal propagation delay and power consumption behavior could take advantage of the aligned version due to the lower resistance and capacitance present in the circuit.

5. CONCLUSIONS AND FUTURE WORKS

A solution to achieve networks with maximal matching between transistor gates was proposed. The algorithms can be applied to any logic style that utilizes disjoint planes. In addition, they can handle transistor network planes containing different amounts of transistors, not being limited to standard CMOS.

Future works will investigate other physical characteristics of transistor networks that could lead to a better layout implementation, taking into account the number of contacts and number of dummy transistors inserted, among other constraints. These research topics will provide knowledge for the future development of an automated tool with the ability of estimating cell layout area.

6. REFERENCES

[1] Marques, F.; Rosa, L.; Ribas, R.; Sapatnekar, S.; Reis, A. DAG Based Library-Free Technology Mapping. Great Lakes Symposium on VLSI 2007 (GLSVLSI'07), pp. 293 – 298.

[2] Mischenko, A. et al. Technology mapping with Boolean matching, supergates and choices. ERL Technical Report, EECS Dept., UC Berkeley, March 2005.

[3] Vujkovic, M. et al. Efficient Timing Closure Without Timing Driven Placement and Routing. Design Automation Conference 2004 (DAC'04), pp. 268 – 273.

[4] Sechen, C.; Guan, B. Large standard cell libraries and their impact on layout area and circuit performance. International Conference on Computer Design 1996 (ICCD'96), pp. 378 – 383.

[5] Stok, L.; Iyer, M.; Sullivan, A. Wavefront technology mapping. Design, Automation and Test in Europe 1999 (DATE'99), pp. 531 – 536.

[6] Correia, V.; Reis, A. Advanced technology mapping for standard-cell generators. Symposium on Integrated Circuits and Systems Design 2004 (SBCCI'04), pp. 254 – 259.

[7] Schneider, F.R.; Ribas, R.P.; Sapatnekar, S.S.; REIS, A.I. Exact Lower Bound for the Number of Switches in Series to Implement a Combinational Logic Cell. International Conference on Computer Design 2005 (ICCD'05), pp. 357 – 362.

[8] da Rosa Jr. L.S.; Marques, F.S.; Schneider, F.; Ribas, R.P.; Reis, A.I. A Comparative Study of CMOS Gates with Minimum Transistor Stacks. Symposium on Integrated Circuits and Systems Design 2007 (SBCCI'07), pp. 93 – 98.

[9] Kagaris, D.; Haniotakis, T. A Methodology for Transistor-Efficient Supergate Design. IEEE Transactions on VLSI Systems, vol.15, n.4, pp. 488 – 492, April 2007.

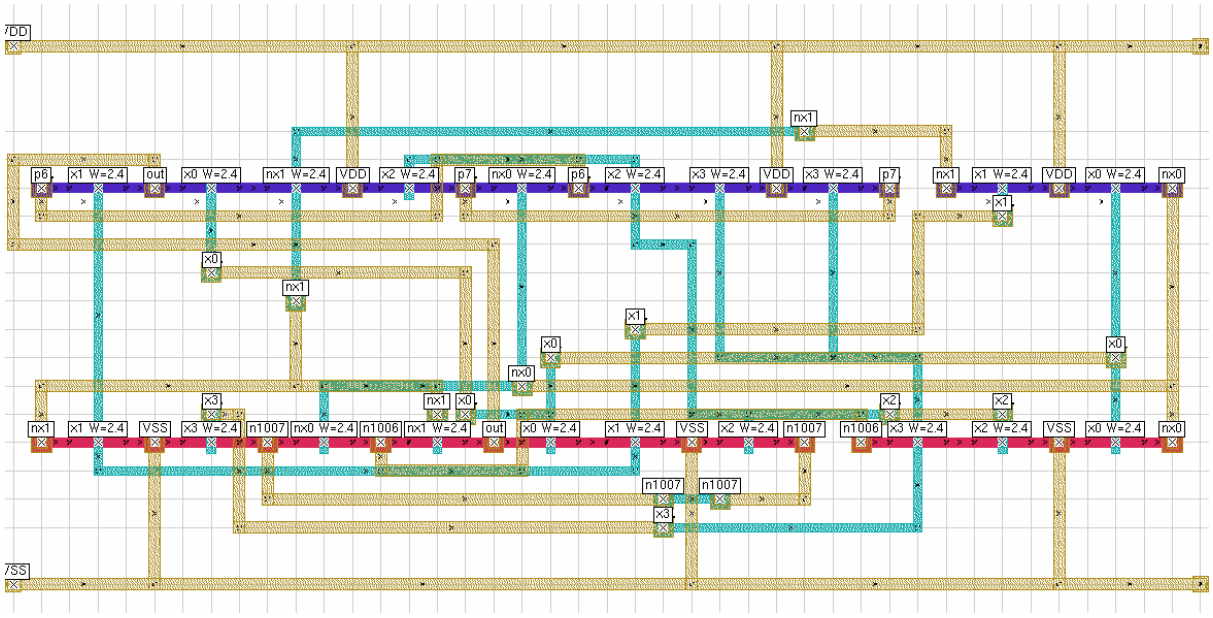
[10] Euler, L. Solutio problematis ad geometriam situs pertinentis. Commentarii Academiae Scientiarum Petropolitanae, vol.8, pp. 128 – 140, 1741.

[11] Even, S. Graph Algorithms. First Edition. Londres: Pitman Publishing Limited, 1979.

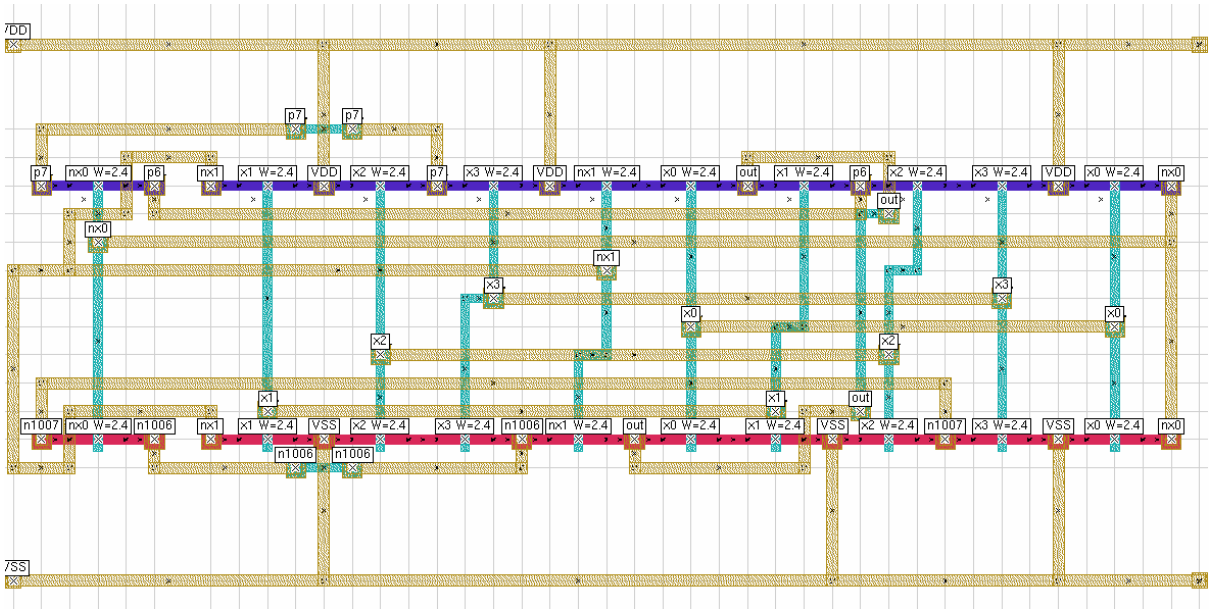
[12] Drozdek, A. Estrutura de Dados e Algoritmos em C++. First Edition. São Paulo: Thomson Learning, 2002.

[13] Uehara, T.; vanCleemput, W. M. Optimal Layout of CMOS Functional Arrays. IEEE Transactions on Computers, vol.c-30, n.5, pp. 305 – 312, May 1981.

[14] Nangate Library Creator™. In: www.nangate.com , June 2007.



(a)



(b)

Figure 4 – Symbolic layouts: (a) unaligned solution, (b) aligned solution.