A New Approach to Optimal Cell Synthesis

Madsen, Jan

Published in:
Proceedings of the IEEE International Conference on Computer-Aided Design, ICCAD-89

Link to article, DOI:
10.1109/ICCAD.1989.76965

Publication date:
1989

Document Version
Publisher's PDF, also known as Version of record

Citation (APA):
A New Approach to Optimal Cell Synthesis

Jan Madsen
DesignCenter of Electronics Institute
Technical University of Denmark
DK-2800 Lyngby, Denmark

Abstract
This paper presents a set of algorithms for optimal layout generation of CMOS complex gates. The algorithms are able to handle global physical constraints, such as pin placement, and to capture timing aspects. Results show that this new approach provides better solutions in area and speed compared to other methods. The algorithms have been implemented in a cell compiler (CELLO) working in an experimental silicon compiler environment.

1 Introduction
Cell compilers, which translate a behavioral cell description into mask layout, makes it possible to deal with much larger "cell libraries" than the usual standard cell approach, by replacing many primitive gates, such as NAND and NOR gates, with a single complex gate [8]. This method reduces layout area and improves performance.

Algorithms pioneered by Uehara and van Cleemput [8], who introduced the idea of Euler paths and dual trails to obtain a layout in the line-of-diffusion layout style, have concentrated on specifying a graph theoretical model of the problem and finding an optimal solution. [5] generalized the work of [8] and proposed an optimal method based upon the same cell model. However they did not address the problem of reordering transistors in the netlist, which may be a considerable limitation as the order influences area as well as electrical performance. [3] propose an algorithm, which does not restrict transistor pairs to be aligned. Though this might have some advantages when trying to chain different cells, the cell height will increase due to a more complex routing within the cell. [2] propose an algorithm based upon the delayed binding concept, which produces a minimum width layout, i.e. an optimal transistor ordering. But as will be shown in the next section, minimum width does not imply minimum area. Common to these algorithms is, that an optimal solution is the one giving the smallest possible cell layout area, i.e. a local optimum.

If, however, a global optimal solution is to be realized, global constraints have to be taken into account, e.g. the pin placement.

So in order to fully utilize the flexibility of the cell compiler concept, it should be possible to handle pin placement as a constraint.

This paper presents a set of algorithms able to handle a constrained pin placement for global optimization and still produce local optimal cell layout if possible.

2 General Design Aspects
Using the line-of-diffusion layout style [8], where physically adjacent transistors may share a common diffusion area, an optimal transistor ordering will maximize the number of shareable diffusion areas and minimize the number of diffusion gaps.

However, a given transistor circuit might have several optimal orderings, all giving a minimal-width layout, but resulting in different heights. Figure 1 shows the layout of three different orderings of the same transistor circuit. Comparing the layout areas, it is obvious that an optimal ordering does not necessarily lead to an area-optimal layout. Further, the choice of ordering significantly influences the circuit performance (i.e. speed) as shown in Table 1, where the worst case propagation delays for the three circuits in Figure 1 are listed.

![Figure 1](image-url)

**Table 1:** Worst case propagation delay (ns) for the circuits from Figure 1.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>1.60</td>
<td>1.66</td>
<td>1.76</td>
</tr>
<tr>
<td>β</td>
<td>0.64</td>
<td>0.73</td>
<td>0.75</td>
</tr>
</tbody>
</table>

In the search for an optimal ordering, the delayed binding concept is often used, [2]. When using this concept, the transistor circuit netlist is regarded as changeable, as long as the function realized by the netlist is not changed. Though the concept of delayed binding may be efficient, one should be able to distinguish...
between fixed and changeable nets/ris, as the ordering also influences the circuit performance. Thus the netlist will be regarded as fixed if pre-timing optimization has been performed. Simulations [4] show that for a medium size complex gate (cf. Figure 3) a worst case delay gain of 22% may be obtained by choosing the right netlist configuration.

The mathematical model should be able to reflect these timing and area aspects, in order to produce a physically correct solution.

3 Model and Problem Formulation

A CMOS transistor circuit at the cell level is composed of a pull-up and a pull-down network. Each of these two networks can be represented by an undirected two-terminal multigraph \( G \), in which an edge \( e \) represents the drain/source connection/path of a transistor and a vertex \( v \) the net connecting several transistors. Each edge is indexed by the gate net (signal) of the corresponding transistor. The two terminals of the multigraph represent the power and output nets.

A transistor ordering, i.e. a path \( P \), which is an alternating sequence of vertices and edges, in which the edges are unique, found by the use of the eulerpath concept, corresponds to unfolding the pull-up and pull-down graphs. Consequently some of the vertices are split into one or more vertices, which has to be connected by internal routing. The way the graph is unfolded and the endpoints of the path are chosen, influences the final layout as shown in Figure 1.

In order to deal with this, all vertices are classified into one of four categories reflecting the characteristics of the layout. A vertex in a two-terminal multigraph is either a terminal-vertex \((T)\) or an internal-vertex \((I)\). If a terminal vertex is connected to the power supply, it is classified as a power-terminal \((P)\), and if it is connected to the output, it is classified as an output-terminal \((O)\). An internal vertex \( v_i \) is classified as either a single-vertex \((S)\) or a multiple-vertex \((M)\), determined by the valence \( v_i \) of the vertex \( v_i \) being equal to or greater than 2.

4 Algorithms

To solve the problems described above, a set of algorithms working on the two-terminal multigraph \( G \) is formulated.

EulerPathExist

This algorithm examines eulerpath existence based on the theorem:

**Theorem 1** A graph \( G \) has an eulerpath iff it is connected and only two vertices (or no vertex) in \( G \) have odd valence. The path \( \{ v_1, e_1, v_2, e_2, \ldots, v_n, e_n, v_{n+1} \} \) is called an open (closed) eulerpath in \( G \) if \( v_1 \neq v_{n+1} (v_1 = v_{n+1}) \).

FindSubGraphs

This algorithm finds the complete set, \( G_{sub} \), of two-terminal serial/parallel subgraphs \( G_{sub} \) in \( G \). Subgraphs are used when reordering the edges in the graph to meet given constraints. The set of subgraphs is deduced by the use of an recursive graph reduction procedure, in which a number of edges are transformed into a single edge representing the subgraph. In each reduction step there are two types of two-terminal subgraphs, serial \( G_{sub} \), and parallel \( G_{sub} \), according to the connection of the edges.

A first-order subgraph is a subgraph not containing any other subgraph, i.e. only edges corresponding to transistors.

4.1 Constrained Pin Placement

If the pin placement is given as a constraint, then depending on whether or not the netlist is fixed, two algorithms can be formulated, which traverse the graphs in order to match the pin placement \( P_{pin} \) to one of the possible eulerpaths.

FindSpecificEulerPath

This algorithm traverses a fixed graph following the path \( P_{pin} \) given by the pin placement. In this case a match of the path to one of the possible eulerpaths cannot be guaranteed. So if a match is not possible, one or more diffusion gaps must be accepted or the given pin placement must be ignored.

ChangeEulerPath

This algorithm performs eulerpath preserving changes on the graph in order to match a given path \( P_{pin} \). The algorithm traverses the graph as in FindSpecificEulerPath. When the next edge from the pin placement can not be reached directly, an edge interchange has to be performed before the path trace can continue. There are two possible interchanges, which do not change the functionality: Edge interchange within a serial two-terminal subgraph \( G_{sub} \) and a two-terminal subgraph swap defined as:

\[
G_{sub}(v_1, v_2) = G_{sub}(v_3, v_4)
\]

In the first case, let \( S_{temp} \) be the set of edges connected to the previous edge \( e_i \). If the next edge \( e_{i+1} \) belongs to a serial two-terminal subgraph \( G_{sub} \), which is connected to \( e_i \), that is

\[
\exists e_m \in S_{temp}, G_{sub} \in S_{sub} : \{e_m, e_{i+1}\} \subseteq G_{sub}
\]

then the edge \( e_{i+1} \) and \( e_m \) are interchanged such that \( e_{i+1} \) will follow the previous edge \( e_i \) as required by the pin placement.

In the second case, let the next edge \( e_i \) does not belong to a serial-subgraph, the problem is far more complicated. First a possible candidate for a subgraph swap has to be found. Depending on whether the path traced so far \( P_{path} \) equals one of the subgraphs or not, there exists two possible ways of choosing a candidate. The path \( P_{path} = \{e_1, e_2, \ldots, e_k\} \), where the edges are defined as \( e_1(v_1, v_2) \) and \( e_{i+1}(v_{i+1}, v_i) \), and let \( S_{sub} \) be the total set of subgraphs in \( G \), then if

\[
\exists G_{sub}(v_1, v_2) \in S_{sub} : (G_{sub} \cap P_{path} = P_{path} \lor P_{path} \cap P_{path} = \emptyset) \land (\{v_1, v_2\} \cap \{v_i, v_j\} \neq \emptyset) \land (\{v_i, v_j\} \cap \{v_{i+1}, v_{i-1}\} \neq \emptyset)
\]

\( G_{sub} \) is a possible candidate for a subgraph swap. The first term selects the subgraph and ensures that the part of the graph which already has been traced will not be changed by the swap, while the second term ensures that \( e_i \) and \( e_{i+1} \) becomes neighbouring edges after the swap. Since the swap should preserve the existence of an eulerpath, it will only be accepted if

\[
f(G_{sub}(v_1, v_2)) = f(G_{sub}(v_3, v_4))
\]

337

Authorized licensed use limited to: Danmarks Tekniske Informationscenter. Downloaded on July 01,2010 at 11:34:27 UTC from IEEE Xplore. Restrictions apply.
lead to swap acceptance. If no swap is accepted, then either the pin placement has to be ignored (terminating the algorithm) or a series of swaps denoted as a multi-swap is performed. A multi-swap is aimed at bringing possible candidates for a swap. The swap is only performed if the output-terminal.

When no pin placement is given the objective is to obtain an area optimal layout, taking into account timing aspects whenever possible.

**4.2 No Constrainted Pin Placement**

When no pin placement is given the objective is to obtain an area optimal layout, taking into account timing aspects whenever possible.

**MakeEulerPathExist**

This algorithm is a special case of ChangeEulerPath, which performs function preserving changes on the graph in order to achieve eulerpath existence. All subgraphs \(G_{sub}(v_i, v_j)\) are possible candidates for a swap. The swap is only performed if the total number of odd valenced vertices are reduced, i.e. if

\[
\exists G_{sub}(v_i, v_j) \in S_{Euler}:
\quad f(v_i) = 1 \land f(v_j) = 1 \land f(z_{v_i}) \neq f(z_{v_j}).
\]

When all subgraphs have been examined, then if at most two vertices have odd valence, the graph has been transformed into having an eulerpath.

**FindEulerPath**

This algorithm finds a eulerpath in the two dual graphs \(G^p\) and \(G^s\) (index \(p \) for parallel and \(s \) for serial) if such exist. Depending on whether or not the netlist is fixed, the algorithm is allowed to rearrange transistors in the two graphs in order to obtain an optimal solution.

First consider the case of a changeable netlist. In a parallel graph the branch to traverse can be selected arbitrarily, the selection corresponds to choose a certain order of elements in the corresponding serial graph. In the case of a changeable netlist the problem can be formulated as to select an order in every serial subgraph which leads to the existence of a dual eulerpath ([8]). Consequently the algorithm starts in the parallel graph \(G^p\), in order to delay the binding of the order of the serial graph.

To reduce the problem size all first-order subgraphs are reduced prior to the path tracing procedure. In \(G^p\) odd parallel and all serial subgraphs are substituted with a single edge, while even parallel subgraphs are removed. In \(G^s\) all first order subgraphs are substituted with single edges. Even serial subgraphs corresponding to even parallel subgraphs in \(G^p\) are marked. The reduced graphs of \(G^p\) and \(G^s\) are denoted \(R^p\) and \(R^s\) respectively.

Two essential properties influence the solution; the choice of endpoints in a closed eulerpath, and at which of the two possible vertices in \(R^p\) to resubstitute a even parallel subgraph. Other algorithms [2] seems to make an arbitrary choice, but as shown in Figure 1, the choice may influence both area and electrical performance. Therefore a set of rules is proposed for an optimal choice of the endpoint in a closed path. In order of preference:

**Rule 1**: Terminal-vertex. Prefer power-terminal to output-terminal. A terminal vertex (especially the power-terminal) will very likely require one routing track. Consequently the splitting of such a vertex will not influence the layout area. Further, the extra amount of diffusion area from the vertex splitting will have the least influence on the electrical performance when added to a power-terminal, while having significant influence when added to an output-terminal (the most sensitive vertex).

**Rule 2**: Internal-vertex. Always choose a multiple-vertex. If a single-vertex is chosen, cell height is increased since the splitting of a single-vertex causes two tracks to be used because of power and output obstructions (cf. Figure 1). Also, the electrical performance is influenced as the parasitic capacitance is significantly increased.

Based upon the theorem:

**Theorem 2** If having two dual graphs \(G^p\) and \(G^s\), a necessary condition for the existence of a dual eulerpath is that the same edges can be reached from endpoints in both \(G^p\) and \(G^s\).
reached directly or by a special marked edge, a gap is inserted. Finally the best solution is chosen. Once a solution has been found all first-order subgraphs can be resubstituted. If possible, high valenced vertices are placed away from the output vertex in order to improve performance.

5 Results

The algorithms described in this paper have been used to implement a cell compiler (CELLO) working in the open experimental silicon compiler environment CATOE

1

[1]. CELLO produces a symbolic layout, which is then transformed into mask layout by a compactor. Figure 3 shows CELLO generated layout after compaction, together with the same circuit realized by two other methods

8

[2] and

2

[2]. In order to make this comparison, the same physical interpretation of the paths was used to generate the layouts for the different algorithms. Table 2 lists the area parameters for different layouts, which shows that the new approach presented in this paper performs better than

8

[2] and

2

[2].

Circuit C3 corresponds to the layouts of Figure 3.

The first test chip generated by means of the CATOE system has been submitted for fabrication in a 2µm n-well CMOS process

2

. Cell synthesis was performed using CELLO. Figure 4 shows a section of the chip core.

6 Conclusion

A new approach for optimal layout generation of CMOS complex gates has been presented. This approach handles global constraints such as pin placement, given by a pre-place and route step, and timing considerations, by distinguishing between fixed and changeable netlists. Further, the approach has been used to implement a cell compiler CELLO, which provides better solutions than previous algorithms.

References


Table 2: Width and height of layouts generated by 3 different algorithms.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>#trans.</th>
<th>Algorithm 1</th>
<th>Algorithm 2</th>
<th>CELLO</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>16</td>
<td>W 77</td>
<td>67</td>
<td>63</td>
</tr>
<tr>
<td></td>
<td></td>
<td>H 71 83</td>
<td>71</td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>12</td>
<td>W 54 63</td>
<td>54</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>H 59 59</td>
<td>59</td>
<td></td>
</tr>
<tr>
<td>C3</td>
<td>14</td>
<td>W 67 67</td>
<td>67</td>
<td>58</td>
</tr>
<tr>
<td></td>
<td></td>
<td>H 71 77</td>
<td>71</td>
<td></td>
</tr>
<tr>
<td>C4</td>
<td>22</td>
<td>W 95 97</td>
<td>97</td>
<td>87</td>
</tr>
<tr>
<td></td>
<td></td>
<td>H 71 95</td>
<td>97</td>
<td></td>
</tr>
</tbody>
</table>

339

Figure 3: Layout results using different algorithms; a) transistor circuit; b) [8]; c) [2]; d) CELLO.

Table 2: Width and height of layouts generated by 3 different algorithms.

1System developed by the CATOE-group (Computer Aided Tool Engineering) at Electronics Institute.

2The chip is produced under the Nordic brokerage service NORCHIP.

Figure 4: Section of the first chip generated by the CATOE system. The individual cells are generated from netlist descriptions by CELLO.