Delay estimation for CMOS functional cells

Madsen, Jan

Published in:
Proceedings of the 2nd European Design Automation Conference

Link to article, DOI:
10.1109/EDAC.1991.206369

Publication date:
1991

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

Delay Estimation for CMOS Functional Cells *

Jan Madsen

Microelectronics Center
Technical University of Denmark
DK2800 Lyngby, Denmark
Email: jan@dc.dth.dk

Abstract

This paper presents a new RC tree network model for delay estimation of CMOS functional cells. The model is able to reflect topological changes within a cell, which is of particular interest when doing performance driven layout synthesis. Further, a set of algorithms to perform worst case analysis on arbitrary CMOS functional cells using the proposed delay model, is presented. Both model and algorithms have been implemented as a part of a cell compiler (CELLO) working in an experimental silicon compiler environment.

1 Introduction

Using *cell compilers*, which translate a transistor netlist (or boolean function) description into mask layout, makes it possible to replace many primitive gates, such as NAND and NOR gates, with a single *complexgate* tuned to the circuit requirements. This methodology leads to a much larger solution space which enables both area and performance driven layout synthesis.

Mapping an optimized function into an appropriate transistor netlist is a one-to-many mapping. The chosen topology will influence both area and performance. Usually performance is optimized at high-level by optimizing boolean expressions or at low-level by transistor sizing. However, choosing the right topology may add yet another performance optimizing step to performance driven layout synthesis. For this purpose we need a simple and efficient model of a general CMOS functional cell which is able to reflect performance properties.

Using the linear *RC* model for modelling digital MOS circuits has become a well accepted practice for estimating circuit delays. This modelling scheme was pioneered by Elmore [1], who's notion of signal delay as the first-order moment of the impulse response has been used widely to approximate the time taken for a signal to reach half of its value.

In 1983, Penfield, Rubinstein and Horowits [2] proposed a method to calculate upper and lower bounds for the delay of a *RC* tree network. This method was extended by Lin and Mead [3] to general *RC* networks and to cover the effect of parallel connections and stored charge, i.e., arbitrary initial charge distribution. Other research on estimating signal delay has been carried out [4], [5].

Though all these methods give good estimates, they are not able to reflect the effect of topological changes within a CMOS functional cell.

In this paper a new *RC* model, called the M model, is proposed. The M model is able to reflect topological changes within a CMOS functional cell and is based upon a detailed study of the impact on circuit delay when changing the order of the transistors in a circuit [6]. Further, we propose an algorithm which uses the M model to estimate worst case delay.

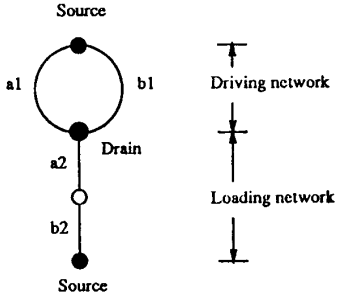

Figure 1: Simple CMOS gate showing "drive" and "load" network when charging output.

2 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 of a transistor and a vertex

*This project has partly been sponsored by the ESPRIT Basic Research Action 3281.

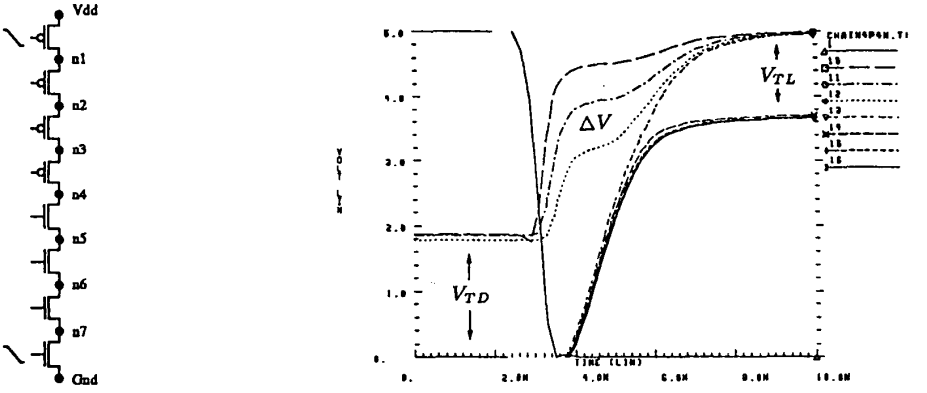

Figure 2: a) Example circuit and b) Voltage of internal nodes during a switch, which charge the output node.

(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 represents the power (source) and the output (drain) (see Figure 1).

In order to estimate worst case delay and switching condition, it is assumed that only one transistor is switching, i.e., all other transistors are either turned ON, in which case they are kept in the graph, or turned OFF and removed from the graph. The graph through which the drain is charged or discharged is denoted the "driving" network. In a CMOS network, a conducting path from source to drain in the driving network (e.g. transistor $b1$ in Figure 1) corresponds to a cutset in the complementary network (e.g. transistor $b2$ in Figure 1), denoted the "loading" network. This cut may still leave transistors in the loading network which has a path to the drain (e.g. transistor $a2$ in Figure 1) and therefore contributes to the charge or discharge of the drain as extra load. Thus, the relation between the two networks depends highly on their topology, i.e., how transistors are arranged, and the model has to handle both networks.

Thus, the problem of finding the switching condition which leads to the worst case delay may be formulated as:

> Find the longest path from source to drain in the driving network, where the transistor next to the source is the one switching, which leads to the highest influence of the loading network.

It is further assumed that the circuit has been stabilized before switching, i.e., that all vertices in the driving network has the same potential and so for the vertices in the loading network.

3 The M Model

Because of symmetry, the following dicussion has been restricted to the case of charging the output through the pull-up network (i.e., the driving network), while the pull-down network acts as the loading network.

Driving Network

The driving network consists of two time components, T_{Drive} and T_{Branch}.

- Driving Path

T_{Drive} accounts for the time used to charge or discharge the output through a series connection of transistors. The calculation is based upon the Elmore time constant. A correction factor of 2 on the first resistance accounts for the fact that the switching transistor is not turned on immediately. T_{Drive} can be expressed as:

$$T_{Drive} = \sum_{i=1}^{n} [a_i \cdot (R_{D1} + \sum_{j=1}^{i} R_{Dj}) \cdot C_i] \qquad (1)$$

where i is the distance from the source, n is the output node and a_i is a correction factor accounting for the different initial charge distribution and the different amount of influence. a_i is defined as:

$$a_i = \begin{cases} \frac{V_{DD} - V_{TD} - i \cdot \Delta V}{V_{DD}} & \text{for } i < n \\ \frac{2}{3} & \text{for } i = n \end{cases}$$

where V_{TD} is the treshold voltage and refering to Figure 2, ΔV is the voltage difference between two neighbouring nodes in the periode where the voltage tend to saturate. ΔV can be expressed as:

$$\Delta V = \frac{V_{DD} - 2V_{TD}}{n+1}$$

- Branches in the Driving Network

Consider Figure 3 showing a path with a branch placed on node i, the influence of the branch is not felt until the next node $i+1$ has to be charged. Therefor the influence of the branch is brought to node $i+1$ rather than node i which is the case for the Elmore model. The contribution

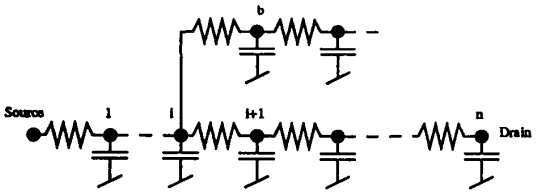

Figure 3: Driving path from source to drain with a branch placed on node i.

to the delay of the driving network by having branches at node i can be expressed as:

$$T_{Branch,i} = c_i \cdot [a_{i+1} \cdot (\sum_{j=1}^{i+1} R_{Dj}) \cdot C_{B_i}] \qquad (2)$$

where c_i accounts for the effect of different initial conditions for the output and the nodes in the loading network. c_i is defined as:

$$c_i = \begin{cases} 0 & \text{if no branch on node } n_i \\ 1 & \text{for } i < n-1 \\ \frac{V_{DD}-V_{TD}}{V_{DD}} & \text{for } i = n-1 \\ \frac{V_{DD}-2V_{TD}}{V_{DD}} & \text{for } i = n \end{cases}$$

I.e. if the branch is added at node $n-1$, the effect is shown at the output node n, but since n initially is at 0 voltage while $n-1$ is at V_{TD}, the branch-capacitance will not be charged until the output reaches V_{TD}. Similar for a branch added at the output, the branch-capacitance will not be charged until the output reaches $2V_{TD}$. In this case $a_{n+1} = \frac{1}{2}$, which reflects the larger sensitivity when adding a capacitance to a loading node.

The branch capacitance C_{B_i} is the capacitance at the first node in the branch counted from the node i. If the branch consists of a large RC-tree, all node capacitances within this tree are pushed to this first node b and the value of the capacitor is corrected according to the little influence when placed far from node i. This situation differs from the loading network in the sense that all nodes in the branch is initially at V_{TD}. The branch capacitance can be expressed as:

$$C_{B_i} = \sum_{nodes} \Delta R \cdot \Delta V \cdot C_{node}$$

where the sum is taken over all nodes in the branch and the two fractions accounts for the different driving resistances and the initial node voltage at V_{TD} respectively:

$$\Delta R = \frac{R_{Path_i} + R_{Db}}{R_{Path_i} + R_{Db} + \sum_{k=b}^{node} R_{Dk}}$$

$$\Delta V = \frac{V_{DD} - (j-1)V_{TD}}{V_{DD}}$$

where j is the distance from node i.

In order to calculate the total contribution from all branches, we add the contribution from all branches along the driving path, i.e., the contribution can be expressed as:

$$T_{Branch} = \sum_{i=1}^{n} c_i \cdot [a_{i+1} \cdot (\sum_{j=1}^{i+1} R_D) \cdot C_{B_i}] \qquad (3)$$

Loading Network

The loading network is a passive RC tree network which increase the load capacitance at the output, thus branches are already included in the model. Since the influence of a capacitance in the loading network depends on how far it is from the output node, a correction factor b_i is introduced. The contribution to the delay from the loading network can then be expressed as:

$$T_{Load} = \sum_{i=0}^{n} b_i \cdot R_{Drive} \cdot C_i \qquad (4)$$

where i is the distance from the output node (i.e. the source) and b_i is the correction factor defined as:

$$b_i = \frac{V_{DD} - V_{TL}}{V_{DD}} \cdot \frac{R_{Drive}}{R_{Drive} + \sum_{k=0}^{i} R_{Lk}}$$

the first fraction accounts for the fact that nodes in the loading network cannot reach V_{DD}, while the last is the fraction between the total resistance driving the output (R_{Drive}) and the total resistance driving node i in the loading network.

Delay Estimate

When the three contributions to the time constant have been found, the delay estimate is calculated as:

$$t_{Estimate} = (T_{Drive} + T_{Branch} + T_{Load}) \cdot \ln 2 \qquad (5)$$

where ln 2 is found from the capacitance charge equation:

$$v_{out}(t) = V_{DD}(1 - \exp^{-t/RC})$$

when the time t is set to the time at wich v_{out} has reached $\frac{V_{DD}}{2}$.

4 Algorithm to Perform Worst Case Analysis

The objective of the this algorithm is to produce a RC tree network from a given CMOS circuit. Both the drive and the load graph (Figure 4a) has to be converted into trees. Figure 4b shows the initial drive and load tree for a

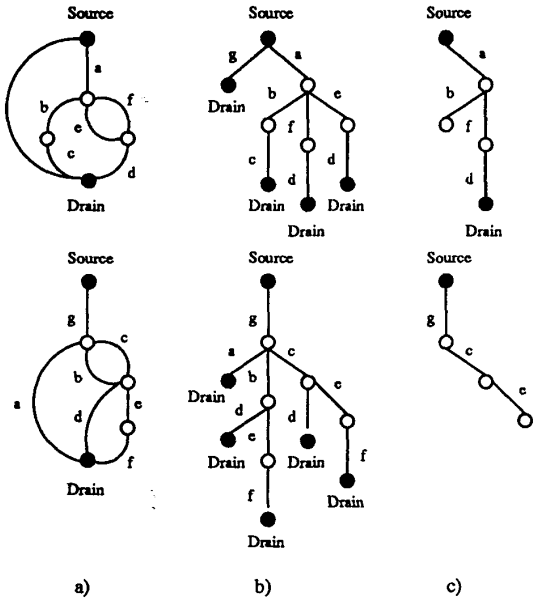

Figure 4: a) drive and load graphs, b) initial trees, and c) final drive and load tree.

complexgate and Figure 4c shows the resulting drive and load tree.

In order to evaluate the time complexity of the algorithms the following properties are defined: Let $| V |$ and $| E |$ denote the number of vertieces and the number of edges in the graph $G(V, E)$; Let $| P |$ represent the number of different paths from source to drain in G (not including loops), and let $| L |$ denote the number of edges in the longest path from the set P of all paths.

LongestPathTree

This algorithm builds a tree of all possible paths from source to drain node in a graph (see Figure 4b). It uses a breadth-first search algorithm which takes $O(| V | + | E |)$ time to construct the tree, using the source node as root. The algorithm terminates when either the drain node has been reached or when edges which can be reached from the present node already is in the path, i.e., avoiding loops. However, loop branches may still be in the tree and have to be removed.

The algorithm is performed on both the drive and the load graph.

RemoveLoopBranches

This algorithm removes loop branches from a tree. For each leaf node which is not a drain node, the algorithm traces from this leaf toward the root. In each step node and connected edge is removed. The trace ends when the current node has more than one edge connected to it. The algorithm takes time $O(| P | \cdot | L |)$.

FindLongestPath

This algorithm creates a list of all paths, from source to drain, in the drive tree, sorted by their length which is calculated using Elmore's time constant. The algorithm takes time $O(| P | \cdot | L |)$.

ReduceByLongestPath

This algorithm is performed for each of the longest paths L_i in the list of paths produced by the FindLongestPath algorithm (e.g., the drive tree of Figure 4b has 3 longest paths). A path L_i from the drive tree, corresponds to a cutset in the load tree. The algorithm removes, from the load tree, all edges belonging to L_i together with all nodes and edges which do not have a path to the root (drain) after the cut, i.e., all nodes and edges which cannot be felt by the circuit output node.

For each leaf node the algorithm traces towards the root, removing the node and connected edge until either the current node is connected to more than one edge, in which case a later trace from another leaf will continue this path, or the current edge belongs to the cutset, in which case the node and edge are removed and the trace is moved to the next leaf node not yet visited. The longest path is then selected as the path L_i from the pathlist which leads to the higest number of remaning edges in the load tree. Thus the time taken by this algorithm is $O(| P_L | \cdot | P | \cdot | L |)$, where $| P_L | (<| P |)$ is the number of longest paths in P.

EvaluateBranches

When the longest path have been found, all other paths have to be cut open. In order to give the highest influence on the delay, these paths are cut as close to the drain as possible. Branches are not placed at the output node (i.e., the drain) as experiments [6] have shown that branches have the least influence when placed at output (and of course at power supply).

For all drain nodes in the drive tree which does not belong to the longest path L, the branch is traced until it intersects with L. The first node (drain) and connected edge are always removed. While either the edge or/and the node has the same identifier as an edge and/or a node in L, the node and connected edge are removed.

Edges which are not removed are included in a cutset, which is used to remove nonconducting edges from the load tree (i.e., open transistors).

As was the case for the previous algorithm, this takes time $O(| P | \cdot | L |)$.

RemoveCutSet

The final algorithm removes all edges belonging to the cutset found in EvaluateBranches. The algorithm also removes all edges (i.e., sub-trees) which are floating, i.e.,

circuit name	# trans. D	# trans. L	M Model delay (ns)	SPICE delay (ns)	% age error
ch1	1	1	0.52	0.59	-11.9
ch2	2	1	1.18	1.18	0.0
ch3	3	1	1.99	1.98	0.5
ch4	4	1	2.96	2.94	0.7
ch5	1	2	0.99	0.95	4.2
ch6	2	2	1.93	1.87	3.2
ch7	3	2	3.02	3.02	0.0
ch8	4	2	4.27	4.39	-2.7
ch9	1	3	1.40	1.18	18.6
ch10	2	3	2.61	2.42	7.9
ch11	3	3	3.97	3.93	1.0
ch12	4	3	5.50	5.65	-2.7
ch13	1	4	1.76	1.31	34.0
ch14	2	4	3.22	2.88	11.8
ch15	3	4	4.85	4.73	2.5
ch16	4	4	6.64	6.79	-2.2
or1	7	7	7.90	7.56	4.5
or2	7	7	7.21	6.70	7.6
or3	7	7	8.17	7.71	6.0
or4	7	7	7.47	6.80	9.9
cap1	4	4	6.64	6.79	-2.2
cap2	4	4	7.82	7.62	2.6
cap3	4	4	9.00	8.43	6.8
cap4	8	6	19.67	21.88	-10.1
cap5	8	6	21.79	23.49	-7.2
cap6	8	6	26.03	26.66	-2.4
cap7	8	6	32.39	31.32	3.4

Table 1: Comparison of the computed delay using the M model and the SPICE (version 3b1) simulations.

edges from which a path to the root cannot be found. This results in the final drive and load tree as shown in Figure 4c.

5 Implementation and Results

The M model and the algorithm set to perform worst case analysis have been implemented in C++ under UNIX as part of the cell compiler CELLO [7]. They have been applied to a large set of benchmarks aimed at showing different aspects of the model and algorithms. Table 1 lists some of the results from the benchmark sets and compare them with SPICE simulation results. All results are the time taken to charge the output node from 0 voltage to half of the power supply. The CPU time taken to analyse and estimate the worst case delays is for the current implementation in the range of $0.1 - 2.1$ sec..

From the table it is seen that the M model compares well with the simulations, except for a feew cases (e.g., ch9, ch13, ch14) in which the load is much larger than the drive, in these cases the effect of the load is to high. The benchmarks or1 - or4 shows the results of different topologies of the same circuit, the M model is able to reflect these topological changes.

6 Conclusion and Future Work

A new *RC*-tree network model for estimating signal delay in CMOS functional cells have been presented. The model is able to reflect topological changes within a cell and compares well with SPICE simulations. Further, a set of algorithms using this model to estimate worst case delay based upon the cell topology has been presented. Both model and algorithm set have been implemented as a part of a cell compiler.

Future work involve refinement of the M model and extension to handle the influence of more than one switching transistor and the influence of different arrival times for the gate signals. These are "global" constraints necessary for choosing the right topology in connection with the surrounding cells.

7 Acknowledgement

The author would like to express special thanks to Dr. Paul Six from IMEC, Belgium, under who's supervision the author, as a guest researcher at IMEC, performed the initial studies leading to the work described in this paper.

References

[1] W. C. Elmore, *"The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers,"* Journal of Applied Physics, 19(1), January 1948, pp.55–63.

[2] Jorge Rubinstein, Paul Penfield, JR. and Mark A. Horowitz, *"Signal Delay in RC Tree Networks,"* IEEE Trans. on CAD, vol. CAD-2, no.3, July 1983, pp.202–211.

[3] Tzu-Mu Lin and Carver A. Mead *"Signal Delay in General RC Networks,"* IEEE Trans. on CAD, vol. CAD-3, no.4, October 1984, pp.331–349.

[4] Pak K. Chan and Kevin Karplus, *"Computing Signal Delay in General RC Networks by Tree/Link Partitioning,"* Proceedings of the 26th ACM/IEEE Design Automation Conference, 1989, pp.485–490.

[5] Serge Gaiotti, Michel R. Dagenais and Nicholas C. Rumin, *"Worst-Case Delay Estimation of Transistor Groups,"* Proceedings of the 26th ACM/IEEE Design Automation Conference, 1989, pp.491–496.

[6] Jan Madsen, *"The Impact of Gate Ordering on Circuit Delay,"* internal paper, Designcenter of Electronics Institute, Technical University of Denmark, EI-LHT148, 27 pages. 1988.

[7] Jan Madsen, *"A New Approach to Optimal Cell Synthesis,"* Proceedings of IEEE International Conference on Computer-Aided Design, Santa Clara, California, 1989, pp.336–339.