CMOS Combinational Circuit Sizing by Stage-wise Tapering

S. Pullela[+], R. Panda, A. Dharchoudhury, G. Vijayan, D. Blaauw
High Frequency Design Methods and Technology, Motorola Inc., Austin, TX
[+] Monterey Design Systems, San Jose, CA

Abstract

We describe a fast (linear time) procedure to optimally size transistors in a chain of multi-input gates/stages. The fast sizing is used in a simultaneous sizing and restructuring optimization procedure, to accurately predict relative optimal performance of alternative circuit structures for a given total area. The idea extends the concept of optimally sizing a buffer chain[5], and uses tapering constants based on the position of a stage in a circuit, and the position of a transistor in a stack.

1. Introduction

Transistor and gate sizing for optimal area/delay trade-off is a well explored problem[1-4]. Sensitivity based iterative approaches and linear/non-linear programming techniques have been proposed. These methods assume a fixed circuit structure and hence become unsuitable for situations, like in [6], where both the structure and the device sizes in a circuit are simultaneously modified for improved optimization.

In [6], a sensitivity-based iterative sizing is done in several steps, while the circuit itself is being restructured between the steps, as shown in Figure 1. The restructuring is done on a set of subcircuits (called windows) which together cover the critical path. This phase involves generating a large number of alternative windows having different logical and transistor level structure than the original windows, trial sizing the circuit with each candidate replacement, and selecting the ones that lead to the best performance for the circuit area at that stage. The dotted curves in Figure 1 arise as alternative windows are inserted at minimum size, and then sized up optimally to current total size. Since several hundreds of structural alternatives may be generated for each window, evaluating all of them by trial sizing the circuit with each one of them will be computationally prohibitive if methods such as [1-4] are employed.

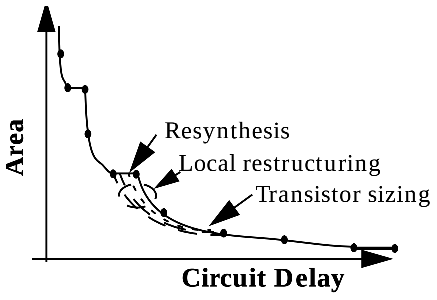

Figure 1: Simultaneous sizing and restructuring

We describe below a fast (linear time) sizing procedure that evaluates alternative structures for their performance at a target total area. On inserting the best structure so predicted, the final sizing for that step can be done by a more accurate method[1-4].

Consider the two alternative 4-input NOR structures shown in Figure 2.

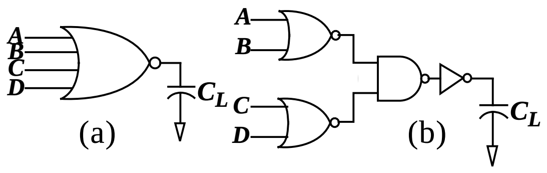

Figure 2: Two alternative 4-input NOR structures

In Figure 3, curves ***Sa*** and ***Sb*** show the optimal sizing trade-off generated by an accurate sensitivity based procedure, for the structures *(a)* and *(b)* in Figure 2, respectively. Note the cross-over point of these curves, demarcating two regions (total area) in which different structures depict better performance relative to the other. Therefore, in comparing two structures for speed at a given area, it is *not necessary* to have their exact sizing trade-off (i.e. curves ***Sa*** and ***Sb***), but rather some approximate sizing behavior that accurately captures the cross-over point would do. This is the basic goal of the fast sizing procedure described below. In fact, the curves ***Ca*** and ***Cb*** in Figure 3 are the optimal trade-off for the respective structures, generated using the proposed fast sizing technique. Note the accuracy with which the cross-over point is predicted.

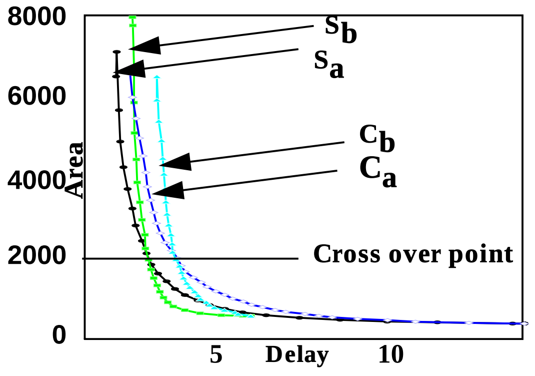

Figure 3: Optimal sizing for two 4-input NOR structures

2. Fast sizing technique

It is shown in [5] that, under certain assumptions, the ratio of sizes of successive stages (called stage tapering ratio) in an optimal buffer chain is constant (equal to *e*), and that the optimal number of stages is uniquely determined by output load and the ratio of drain to gate capacitances of a unit buffer. The consequence of this result is that the task of sizing the buffers in a chain (presumably with optimal number of stages) reduces simply to one of allocating the total area among the stages based on the tapering ratio, without a need to consider sensitivities. Our main idea is to extend this basic result to a chain of multi-input gates/stages of arbitrary length, so that optimal sizes of stages, and optimal sizes of transistors within stages, can be easily determined through allocation of the total area, using certain tapering ratios. Several modifications, however, are necessary to account for the deviations from the simpler buffer chain situation.

2.1 Non-linear tapering factor

When the number of stages in a buffer chain is fixed, and assumptions of [5] are not valid, the optimal tapering ratio is no longer a constant. We have found that, for a given total area, the optimal tapering ratio K_i (relating total sizes of *i-th* and *i+1-th* stages) is a stage-dependent non-linear function: $K(i, N, C_L)$, where i is the position of the stage w.r.t. input/output, N is number of stages, and C_L is output load. For a given technology, we determine this function through sensitivity based sizing of a large number of inverter chains, varying N and C_L, and constructing a table, for a one-time cost.

2.2 Stack depth factor

A 3-input NAND requires more than 3 times the area of an inverter of comparable worst case transition delay. Intuitively, therefore, stages with larger stack depths in a chain would require relatively larger area allocation to preserve the optimality of the sequence. Thus, to extend the idea of optimal buffer/inverter chain to a chain of multiple input stages, we multiply the stage tapering factor, K_i, by a stack depth factor, d_f (>1), defined as the ratio of areas required for an n-input stack to yield the same delay as an inverter. Assuming a constant stack tapering ratio r, we can show that the Elmore delay of an n-input stack is:

$$D(n) = H_1 r^n / A_n + n H_2 C_N / A_n \qquad r>1 \qquad (1)$$

where H_1 and H_2 are technology parameters, A is the stack area, and C_N is the next stage capacitance. By requiring $D(n)$ equal $D(1)$, we can get the d_f factor, which equals A_n/A_1, as a function of r. The values of r that minimize d_f are pre-determined for different stack types and stack depths.

From (1), it appears that we need to know the capacitance that this stage drives to determine d_f. This is not a problem if we walk from outputs back to inputs to determine the stage tapering constants, as stage capacitance is proportional to the stage tapering constant.

2.3 Position Modifier

To compute the relative sizes of the transistors within a stack, we also define the *position modifier* p_m of a transistor, which is simply a function of r, the stack depth n, and the position, h, of the transistor in the stack.

$$p_m = r^h (1-r)/(1-r^n) \qquad (2)$$

Based on the above factors, the optimal size of a transistor is taken to be proportional to a weight, $\mathbf{w_m = p_m\, d_f\, K_i}$. To resize a sequence of stages, the sizing algorithm walks from outputs to inputs, determining K_i and d_f for each stage, and thereby w_m for each transistor. The total target area is now allocated among the transistors based on their weights.

3. Results and conclusion

We implemented this procedure in [6], to replace the sensitivity -based sizing during the window selection process. In order to test this method, we ran this algorithm, and also the detailed sensitivity based algorithm, on several windows generated during resynthesis. Typically each of these windows had about 20 gates. Each resynthesis step generated about 50-100 windows depending on the size of the circuit. Out of a total of approximately 600 windows, in about 95% of the cases, our method was able to identify the correct relationship between two windows at the given area.

Table 1: Results of Resynthesis with Fast Sizing.

		Run Time (Hrs.)		Acceptance Rate %	
Ckt	**size #trans.**	**accurate sizing**	**fast sizing**	**no filtering**	**with filtering**
MSTSLV	**445**	**6.67**	**0.92**	**24**	**60**
PDEC	**2815**	**24**	**3**	**12**	**40**

In Table 1, Columns 3 and 4, we show the improvement in run time taken to generate the entire trade-off for the circuits, with the aid of fast sizing. Run time improved significantly (6X to 8X), with no performance penalty.

In Column 5, we show the percentage of windows that resulted in actual performance improvement, and hence accepted, when sized up to the target total area,. This is in comparison to the number of windows generated at the restructuring step that looked promising based on the evaluation at minimum window area (corresponding to minimum transistor sizes in the window). The poor acceptance rates indicate the amount of wasteful computation of the costly iterative sizing techniques, spent on the rejected windows. The figures in column 6 show tripling of the acceptance rate when a filtering of candidates was done using the fast sizing technique. The additional time spent in filtering is more than compensated for, by saving the computations of trying out bad choices. Of course, one would expect a much higher acceptance rate given a 95% success in identifying the correct windows. However, when the windows were inserted back into the original circuit, and the entire circuit was sized, we found that parts of the circuit outside the window were sized resulting in the window not being sized to the area at which it was evaluated.

In general, we have observed that while about 80% of the new windows were found to be better at minimum size, only about 20% of the windows were acceptable at the target area. Our sizing method clearly identified these windows in a fraction of the time it would take for a sensitivity based sizing algorithm. As a matter of fact, since structural deficiencies were not compensated for, unlike in sensitivity based methods, we observed that the differences were amplified, compared to the standard sizing techniques.

4. References

[1] Fishburn, J. P, et. al., "TILOS: A Posynomial Programming Approach to Transistor Sizing," *ICCAD*, Nov 1985, pp269-283.

[2] S. S. Sapatnekar, et. al., ``An Exact Solution to the Transistor Sizing Problem for CMOS Circuits using Convex Optimization," *IEEE Trans. on CAD*, Vol. 12, No. 11, pp. 1621 - 1632, November 1993.

[3] Noel Menezes, et. al., "A Sequential Programming Approach to Concurrent Gate and Wire Sizing," *ICCAD*, Nov 1995, pp 144-151.

[4] Berkelaar, et. al., "Computing the Entire Area Delay Trade-off Curve for Gate Sizing with a Piecewise Linear Simulator," *ICCAD*, 1997, pp 474-480.

[5] Shoji. M, "Theory of CMOS digital circuits and circuit failures," *Princeton University Press, 1992.*

[6] S. Gavrilov et al., "Library-Less Synthesis for Static CMOS Combinational Circuits," *ICCAD*, Nov 1997, pp 658-662.