Gate Size Optimization for Row-based Layouts

Naresh Maheshwari Sachin S. Sapatnekar
Department of Electrical and Computer Engineering
201 Coover Hall, Iowa State University, Ames, IA 50011, USA.
{naresh,sachin}@iastate.edu

Abstract

A transistor sizing algorithm for row-based layouts is presented under a improved area model. This algorithm uses convex programming to find a minimal area circuit for a given delay specification. The new area model uses a concept of row heights as opposed to the conventional metric of sum of gate sizes. Results over a number of circuit indicate a significant reduction both in the minimum delay achievable and area as compared to TILOS-like optimizer.

1 Introduction

One of the primary design objectives while synthesizing CMOS circuits is to reduce both the area and the delay. These two requirements directly contradict each other, and the tradeoff involved in developing an acceptable compromise constitutes the sizing problem. Most CMOS circuits are sequential circuits that consist of combinational subcircuits between clocked registers. Since the clock period is determined by the maximum delay over all such combinational subcircuits, a popular formulation of the optimization problem is to minimize the area of each combinational subcircuit subject to delay constraints.

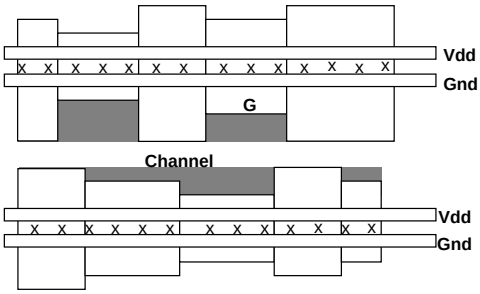

Figure 1: Macrocell Layout using a Row-based Style

Previous methods have estimated the area of a layout as the sum of all transistor sizes. In many layouts, however, including macromodule based layouts [1] and many custom layout styles, transistors are organized in rows with channels left between rows for routing. In such ordered layouts, the height of a row is governed by the maximum height of a cell within the row, which may be variable if the cells in a row are sized differently. This is illustrated in the layout shown in Figure 1, taken from the layout design system in [1]. The height of each rectangular row is given by the maximum of the cell heights. Under a structured layout style such as that in [1], the height of each cell is proportional to the gate size, and the maximum gate size in each row determines the contribution of that row to the area of the layout. This is because most of the CAD tools for channel routing in industry work with rectangular shapes, and even in the presence of over-the-cell routing, the space corresponding to the shaded region in Figure 1 is largely wasted, and can be used for sizing without any detriment to the total area of the layout. In the figure, an increase in the size of the gate G incurs no additional area penalty, and may serve to reduce the circuit delay.

The motivation for this work is provided by the improvements in accuracy that are provided by a more accurate area measure, and the exploitation of this measure to produce more compact layouts. It is shown that the problem of sizing row-based layouts is equivalent to a convex programming problem, and an exact algorithm for convex programming [2] is used to find a solution.

2 Area and Delay Modeling

Area Modeling: For the purposes of this work, we consider gates to be laid out using the style in [1], where variable-height cells have been used in a row-based layout scheme.[1] Under this layout style, the height h_i of a gate G_i is given by $h_i = w_{n,i} + w_{p,i} + K$ where $w_{n,i}$ is the width of its n transistors, $w_{p,i}$ the width of the p transistors, and K is a constant that accounts for the space needed to connect p-diffusion and n-diffusion, etc. The length of the gate depends on the number of inputs and is independent of the transistor sizes.

The height of the i^{th} row, T_i, is determined by the height of the tallest gate in the row, $T_i = max_{j \in row_i}(h_j)$. The total area of the chip with n rows

[1] It is possible to adapt the algorithm to other layout styles too.

is then given by

$$A = \left[\max_{\forall j} L_j\right] \cdot \sum_{j=1}^{n} T_j \qquad (1)$$

where L_j is the length of row j. Under the design style used, the length of a row will not change since changing gate sizes will only result in a change in the cell height. Therefore, the objective of minimizing the total area is the same as minimizing the sum of the T_i's. It is assumed here that the transistor sizing tool will not be allowed to change the placement of gates in rows. An iterative design procedure would go back and forth between placement and sizing until an optimal result is obtained.

Delay Modeling: We first show how an n transistor is modeled by a set of capacitances and resistors. A p transistor of width $w_{p,i}$ is similarly modeled. Since all the transistors are set to minimum length, the capacitances can be modeled in terms of only the transistor widths. For an n transistor of width $w_{n,i}$ [2], we can write the source/drain capacitance $C_{sdn_i} = C_{d,n_1} \cdot w_{n,i} + C_{d,n_2}$, and the gate capacitance $C_{gn_i} = C_{g,n_1} \cdot w_{n,i} + C_{g,n_2}$ where $C_{d,n_1}, C_{d,n_2}, C_{g,n_1}$ and C_{g,n_2} are constants. The on-resistance, $R_{i,n}$, of an n transistor is given by $R_{i_n} = \frac{R_n}{w_{n,i}}$.

At the gate level, each gate is modeled by an equivalent inverter, parameterized with all n (p) transistor sizes set to $w_{n,i}$ ($w_{p,i}$). In this implementation, only static CMOS gates are considered. All transistors of the same type in a gate are assumed to have a uniform size. The pull-up (pull-down) structure is represented by an equivalent inverter with p transistor (n transistor) size of $S_{p,i}$ ($S_{n,i}$); this number is referred to as the gate size. The relation between the gate sizes in the equivalent inverter and transistor widths in the gate can easily be computed for various type of gates.[2] The capacitance loading, C_{L_i}, of a gate G_i can be calculated from the transistor sizes of its fanouts as follows:

$$C_{L_i} = \textstyle\sum_{j \in fanout_i}(C_{gn_j} + C_{gp_j}) + C_{intrinsic} \qquad (2)$$

where $C_{intrinsic}$ corresponds to the source and drain capacitance connected to the output node of G_i.

The Elmore fall delay, (t_{f_i}), of gate G_i can then be obtained from C_{L_i} and $S_{n,i}$ as $t_{f_i} = \frac{R_n \cdot C_{L_i}}{S_{n,i}}$. The rise delay is similarly obtained as $t_{r_i} = \frac{R_p \cdot C_{L_i}}{S_{p,i}}$. The PERT procedure [3] is used to find the critical path in the circuit. The implementation of this procedure is as in [2].

[2]For example, for a k-input NAND gate, $S_{n,i} = w_{n,i}/k$, $S_{p,i} = w_{p,i}$). Notice that $S_{p,i}$ is $w_{p,i}$ (and not $k \cdot w_{p,i}$) since in the worst case, only one of the k transistors in parallel will be on.

3 Optimization Problem Formulation

For a combinational circuit, the transistor sizing problem is formulated as the problem of minimizing the circuit area subject to delay constraints as follows [4, 2]:

$$\begin{array}{rcl} minimize & Area & (3) \\ subject\ to & Delay \leq T_{spec} & \\ and & \text{Each transistor size} \geq \text{Minsize} & \end{array}$$

Previous approaches have modeled the area as the sum of the transistor sizes, and it is shown that the problem is equivalent to a convex programming problem. In row-based layouts, however, a more appropriate area model is the one described in Section 2.1. We proceed to show now that the convexity properties continue to hold under this area model.

The delay model here models the circuit delay using posynomial functions [5] of the gate sizes, as has been done before. If each gate size x_i is made to undergo the transformation $x_i = e^{z_i}$, then the feasible set is defined by a convex region in the space of z_i's [4, 2]. As shown in Section 2.1, minimizing the area of a layout with n rows involves the minimization of

$$\sum_{i=1}^{n} \max_{j \in row_i} (w_{n,j} + w_{p,j}) \qquad (4)$$

The objective is a sum of the maximum of posynomial functions in the gate sizes, and is transformed using the mapping above to the maximum of convex functions of the gate sizes. Since the sum and the maximum of a finite number of convex functions is convex, the objective is a convex function, to be minimized under constraints specified by a convex set. Therefore, the problem has been shown to be a convex programming problem. As a consequence, any local minimum is a necessarily a global minimum.

The optimization algorithm used here is an efficient convex programming algorithm. Details about the optimizer are provided in [2], and not described here due to space limitations. The complexity of the algorithm is $O(n^{2.5})$, where n is the number of variables. The algorithm requires the computation of the area gradient and the delay gradient of all the pull-up and pull-down structures. Efficient algorithms for the computation of these gradients have been embedded into the implementation.

4 Experimental Results

The algorithm is implemented in C as a program, **Rowsize**, that constitutes about 1200 lines of code. Although most ISCAS89 benchmark circuits were

tested, due to limited space only a few are presented here. The circuit delays are in nanoseconds while the execution times on a DEC ALPHA 3000/600 workstation are in seconds. As mentioned earlier the area is measured as the sum of row heights. Since the ISCAS89 benchmarks did not contain any information about the layout, the gates were assigned arbitrarily to rows. The unsized area and delay correspond to circuits with all there transistors set to minimum width of one. Note that since height of each row in a unsized circuit is 2 units (height $= W_n + W_p = 1 + 1 = 2$), the unsized area is twice the number of rows.

The obtained results were compared with our implementation of the TILOS algorithm from [4], where the objective function is to minimize the sum of all transistor sizes (note that this is dissimilar from the objective function for Rowsize).[3] After performing this optimization, the area of the resulting row-based layout is measured under our more accurate area model. The comparison is shown in Table 1 and Figures 2-7. In Table 1, D_u and A_u refer to the delay and area, respectively, of the minimum-sized circuit. The number of gates, G, in each circuit are also shown. For several values of the delay specification, the area of the optimum circuit as calculated by Rowsize and the TILOS-like optimizer are shown. The execution time, T_{ex}, for each algorithm is also shown. A "-" in Table 1 implies that our implementation of TILOS could not achieve the delay specification. Although it is seen that the CPU times required by Rowsize are typically much larger than those for the TILOS-like optimizer, it can be seen that Rowsize gives far superior results for this area model.

It is observed from the results that for loose delay constraints there is not much of a difference between TILOS and the convex optimizer. As the delay constraint get tighter, the convex optimizer performs much better than the TILOS-like optimizer in terms of area of the circuit for similar delays. In all cases the convex optimizer is able to achieve much smaller delays than the TILOS-like optimizer.

In a small number of cases, for loose constraints Rowsize takes slightly more area than TILOS-like optimizer. This can be attributed to the use of loose convergence parameters.

5 Conclusion

In this paper, we have presented a convex programming approach with a new area model that is particularly suitable for row-based layouts, e.g. macro-based or channel routing based layouts. We have also demonstrated that considerable reduction can be obtained both in the area and the minimum delay achievable in comparison with the TILOS algorithm. From our results, we conclude that the TILOS algorithm provides good results for loose constraints, and has a low run-time. For tighter constraints, however, the convex programming algorithm provides significant improvements although it is computationally expensive. Therefore, it is a good idea to use TILOS under very loose constraints, and the convex programming method for critical subcircuits.

[3] We tried to adapt the TILOS-like optimizer to minimize the objective function for the row-based layout problem directly. However, the heuristic does not work well for this objective function.

References

[1] J. Kim, S. M. Kang, and S. S. Sapatnekar, "High performance CMOS macromodule layout synthesis," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, pp. 4.179–4.182, 1994.

[2] S. S. Sapatnekar, V. B. Rao, P. M. Vaidya, and S. M. Kang, "An exact solution to the transistor sizing problem for CMOS circuits using convex optimization," *IEEE Transactions on Computer-Aided Design*, vol. 12, pp. 1621–1634, Nov. 1993.

[3] T. Kirkpatrick and N. Clark, "PERT as an aid to logic design," *IBM Journal of Research and Development*, vol. 10, pp. 135–141, Mar. 1966.

[4] J. Fishburn and A. Dunlop, "TILOS: A posynomial programming approach to transistor sizing," in *Proceedings of the IEEE International Conference on Computer-Aided Design*, pp. 326–328, 1985.

[5] J. Ecker, "Geometric programming: methods, computations and applications," *SIAM Review*, vol. 22, pp. 338–362, July 1980.

Table 1: Comparison of Rowsize vs TILOS

Circuit	Delay	*Rowsize*		*Tilos*		Circuit	Delay	*Rowsize*		*Tilos*	
	Spec	Area	T_{ex}	Area	T_{ex}		Spec	Area	T_{ex}	Area	T_{ex}
D_u in ns	ns		sec		sec	D_u ns	ns		sec		sec
i135	33.6	4.22	6871	5.79	1.2	**frg1**	20.8	4.2	256	7.4	0.8
$G = 269$	29.9	4.47	3163	214.7	36	$G = 109$	19.8	4.41	144	103	69
$D_u = 39.4$	27.9	4.70	2489	-	-	$D_u = 21.8$	14.6	6.6	71	-	-
$A_u = 4$	10.7	73.3	287	-	-	$A_u = 4$	11.5	55.0	230	-	-
f51m	20	8.3	616	9.1	0.5	**cm150a**	18.9	4.5	66	8.4	0.8
$G = 136$	17.2	8.7	430	53	99	$G = 72$	18.3	4.7	69	53.9	35.7
D_u =22.4	13.8	10.6	238	-	-	$D_u = 21.5$	15.5	7.2	33	-	-
$A_u = 8$	10.2	84.9	1213	-	-	$A_u = 4$	12.9	69.7	18.8	-	-
count	33.1	26.0	250	24.8	0.2	**cm151a**	32	4.06	14	7.3	0.2
$G = 144$	30.5	26.8	133	769.0	24	$G = 34$	30.6	4.13	8.5	103	44
$D_u = 35.5$	19.4	42.7	44	-	-	$D_u = 32.7$	16.7	5.99	3.2	-	-
$A_u = 24$	12.5	284	13.2	-	-	$A_u = 4$	9.8	55.7	1	-	-

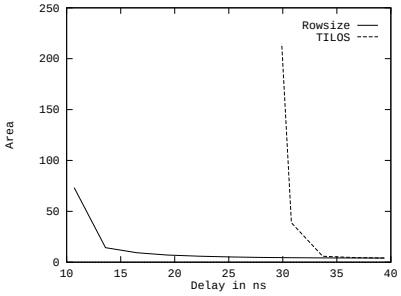

Figure 2: Area-delay Curve for i135

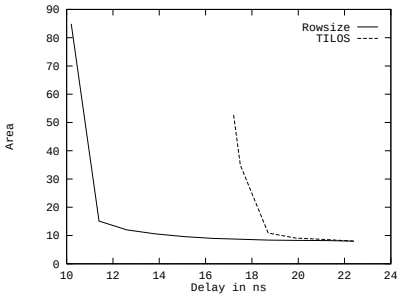

Figure 3: Area-delay Curve for f51m

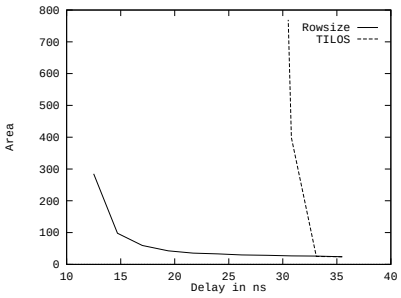

Figure 4: Area-delay Curve for count

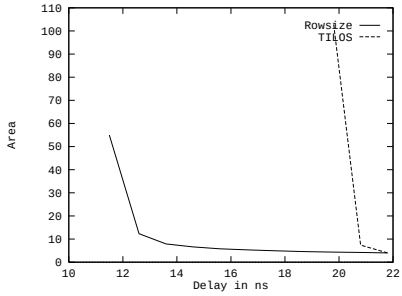

Figure 5: Area-delay Curve for frg1

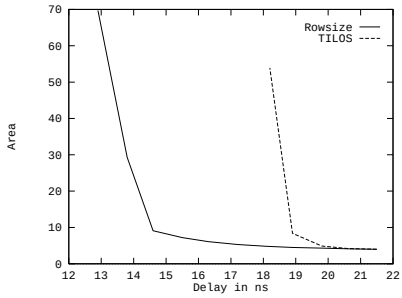

Figure 6: Area-delay Curve for cm150a

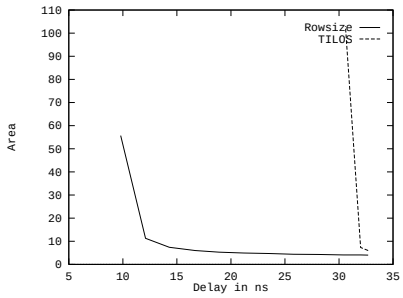

Figure 7: Area-delay Curve for cm151a