Improving Placement under the Constant Delay Model

Kolja Sulimma[1], Ingmar Neumann[1], Lukas van Ginneken[2], Wolfgang Kunz[1]

[1]EDA Group, Department of EE and IT
University of Kaiserslautern, Germany

[2]Magma Design Automation, Inc.
Cupertino, CA, USA

Abstract

In this paper, we show that under the constant delay model the placement problem is equivalent to minimizing a weighted sum of wire lengths. The weights can be efficiently computed once in advance and still accurately reflect the circuit area throughout the placement process. The existence of an efficient and accurate cost function allows us to directly optimize circuit area. This leads to better results compared to heuristic edge weight estimates or optimization for secondary criteria such as wire length.

We leverage this property to improve a recursive partitioning based tool flow. We achieve area savings of 27% for some circuits and 15% on average. The use of the constant delay model additionally enables timing closure without iterations.

1 Introduction

Modeling the delay of a real CMOS gate is a complex task. For a given gate realization the delay is a non-linear function of many parameters: the load capacitance, for example, has a major effect on the delay. For longer interconnects resistive effects must be considered and even inductance might play an increasingly important role. Signals on adjacent wires can influence the delay by cross coupling and forward coupling makes the delay dependant on the slew of the input signal. Local heating could slow down a gate and in SOI systems the delay even depends on the signal history.

To reflect these complex dependencies some cell library vendors choose not to characterize their cells by abstract models but use multi-dimensional tables. Timing simulators then interpolate between the table entries.

Models like these, however, are too irregular to be suitable for optimizations early in the design process. Instead, the cell delay usually is approximated by a simplistic model that can be used to efficiently guide the synthesis process. For example, it is very common to model the delay d_i of a gate i as a linear function of the lumped load capacitance C_i:

$$(1)\quad d_i = d_i^* + \partial d_i / \partial C_i \cdot C_i$$

The factor $\partial d_i / \partial C_i$ describes the dependency of the delay of gate i on the load C_i present at the output of i. d_i^* represents the load independent portion of the gate delay.

1.1 The Constant Delay Model

A different approach can be taken if the cell library provides multiple cell sizes for each logic function or if cells can be generated on the fly by a cell generator [8]. For a given – and realizable – cell delay and a given cell environment we can find a cell size that approximately realizes the desired delay. In these cases the cell delay is constant and the required cell area is a function of the load capacitance and all the other parameters mentioned above.

If the cell's circuit topology is unchanged for the various cell sizes the cell area can be approximated as a linear function of the lumped load capacitance:

$$(2)\quad A_i = A_i^* + \partial A_i / \partial C_i \cdot C_i$$

This is called the constant delay model [6]. Analogous to Equation (1) the factor $\partial A_i / \partial C_i$ describes the dependency of the area of gate i on the load C_i present at the output of i. A_i^* represents the load independent portion of the gate area.

The load capacitance C_i in Equation (2) consists of the input capacitances C_{ij} of the successor gates j driven by i and the capacitance of the net connecting them.

$$(3)\quad C_i = C_i^{wire} + \sum_{\text{successors } j} C_{ij}$$

Resizing a gate also changes its input capacitance. For each input k of a gate i the capacitance is modeled by a base capacitance C_{ki}^* and a factor $\partial C_k / \partial C_i$ describing the dependency of the capacitance of the gate input connected to node k on the load capacitance present at the output of i. The actual input capacitance C_{ki} for a load C_i results to:

$$(4)\quad C_{ki} = C_{ki}^* + \partial C_k / \partial C_i \cdot C_i$$

For simplicity, in the following, the net driven by gate i is also denoted i.

It should be mentioned that like all simplistic delay models the constant delay model is only an approximation. Most notably, the cell sizes are quantized and there is a minimum size for a cell. The quantization is coarser for cell libraries than for cell generators and in cell libraries there also is a maximum size for a cell.

2 Circuit Area in the Constant Delay Model

2.1 Area sensitivity in combinational circuits

When using the constant delay model the circuit delay is fixed and is therefore not an optimization objective. Instead, the area necessary to realize the required circuit delay is subject to minimization efforts.

The total area of a constant delay model netlist N_C is the sum of the individual gate areas.

$$(5)\quad A = \sum_{i \in N_C} A_i$$

The area A_i of an individual gate i depends on the capacitance C_i on the gate's output. Unfortunately, C_i depends on the size of its fanout gates by Equations (3) and (4). These gate sizes in turn depend on their load capacitance and so on.

However, it is possible to express how the capacitance at some node i influences the total circuit area A. We denote this global area sensitivity with $\partial A / \partial Ci$ in correspondence to the area sensitivity $\partial Ai / \partial Ci$ of a single gate. The values of $\partial A / \partial Ci$ can be efficiently computed for combinational circuits in linear time as follows.

The global area sensitivity of the circuits driving the primary inputs must be known in advance. Alternatively, they can be assumed to be zero. This is equivalent to assuming that the primary inputs are driven by drivers of infinite strength. This assumption is commonly made for timing analysis.

Now consider the gates of the circuit in topologically sorted order. Adding capacitance to the output node of gate i will increase its area by $\partial Ai / \partial Ci$. But also its input capacitances increase and contribute to the capacitance seen by its predecessors k. Fortunately, we already know how these capacitances affect the circuit area. This allows us to compute the global area sensitivity of node i as

$$(6)\quad \partial A / \partial C_i = \partial A_i / \partial C_i + \sum_{\text{input } k \text{ of } i} \partial A / \partial C_k \cdot \partial C_k / \partial C_i$$

This means that not only the area of a single gate is increasing linearly with the load on a single net but also the total area A does. An increase of C_i by ΔC, for example, will increase A by $\partial A / \partial Ci \cdot \Delta C$.

The circuit area A^* in the absence of wiring capacitance can be easily computed by iteration from the outputs to the inputs. If we denote the wiring capacitance at node i with C_i^{wire} we obtain the circuit area as a weighted sum of wiring capacitances.

$$(7)\quad A = A^* + \sum_{i \in N_C} \partial A / \partial C_i \cdot C_i^{wire}$$

2.2 Area sensitivity in sequential circuits

The simple and efficient method to calculate the global area sensitivities presented in the previous section only applies to combinational circuits that can be topologically sorted. Sequential circuits require a more complicated analysis.

Merging Equations (3) and (4) results in an equation describing the capacitance C_i seen at a node i as a function of the capacitances at the successor nodes.

$$(8)\quad C_i = C_i^{wire} + \sum_{\text{successors } j} \left(C_{ij}^* + \partial C_i / \partial C_j \; C_j \right)$$

The equations for all nodes of the circuit form a set of linear equations that can be formulated in matrix notation.

$$(9)\quad \bar{c} = \bar{w} + \bar{c}^* + D \cdot \bar{c}$$

Equation (9) follows from Equation (8) by using the following substitutions: D is the matrix of local sensitivity factors $\partial Ci / \partial Cj$. Factors $\partial Ci / \partial Cj$ that do not exist in the circuit are set to zero.

$$D = \left(\partial C_i / \partial C_j \right)_{\substack{1 \le i \le n \\ 1 \le j \le n}}$$

Vector $\bar{w}$ contains in row i the wiring capacitance at node i.

$$\bar{w} = \left(C_i^{wire} \right)_{1 \le i \le n}$$

Vector $\bar{c}^*$ contains in row i the gate size independent portion of the input capacitances of the successors.

$$\bar{c}^* = \left(\sum_j C_{ij}^* \right)_{1 \le i \le n}$$

$\bar{c}$ is the vector of node capacitances.

$$\bar{c} = \left(C_i \right)_{1 \le i \le n}$$

After some reordering of Equation (9) to

$$(10)\quad (D - I) \cdot \bar{c} = -(\bar{w} + \bar{c}^*)$$

we can multiplying by the inverted matrix $(D-I)^{-1}$ and obtain an equation to compute the capacitances if we know the wire lengths:

$$(11)\quad \bar{c} = -(D-I)^{-1} \cdot (\bar{w} + \bar{c}^*)$$

The area of a gate A_i is a linear function of a node capacitance as shown in Equation (2). The total circuit area is the sum of all gate areas and can therefore be formulated as scalar product of the vector $\bar{c}$ and the vector of all area sensitivities.

$$A = \bar{c} \cdot \left(\partial A_i / \partial C_i \right)_{0 < i \le n} + A^* = -\left((D-I)^{-1} \cdot (\bar{w} + \bar{c}^*) \right) \cdot \left(\partial A_i / \partial C_i \right)_{0 < i \le n} + A^*$$

Using the definitions above this expands to

$$(12)\quad A = -\sum_i \left(\sum_j \left((D-I)^{-1} \right)_{i,j} \cdot \left(C_j^{wire} + \sum_k C_{jk}^{*} \right) \right) \cdot \partial A_i / \partial C_i + A^{*}$$

The global area sensitivities are the partial differential coefficients of Equation (12)

$$(13)\quad \partial A / \partial C_i^{wire} = -\sum_j \partial A_j / C_j \left((D-I)^{-1} \right)_{ij}$$

2.3 Example

Figure 1 shows a combinational circuit example with three constant delay cells. Within the boxes the cell parameters affecting the example are shown. Starting from the inputs the area sensitivities of all nets can be computed.

The fanin of net 4 consists only of gate 4 itself. Therefore, the overall area sensitivity $\partial A/\partial C4$ of net 4 is equal to the area sensitivity $\partial A4/\partial C4 = 2$ of gate 4.

Adding one unit load to net 5 increases the size of gate 5 by $\partial A5/\partial C5 = 3$ area units. This adds $\partial C4/\partial C5 = 0.4$ units of extra load to net 4 causing gate 4 and its fanin to increase in size by $\partial C4/\partial C5 \cdot \partial A/\partial C4 = 0.4 \cdot 2 = 0.8$ area units. As a result we get that $\partial A/\partial C5 = 3.8$. One extra unit load capacitance on net 5 thus will increase the total circuit area by 3.8 units.

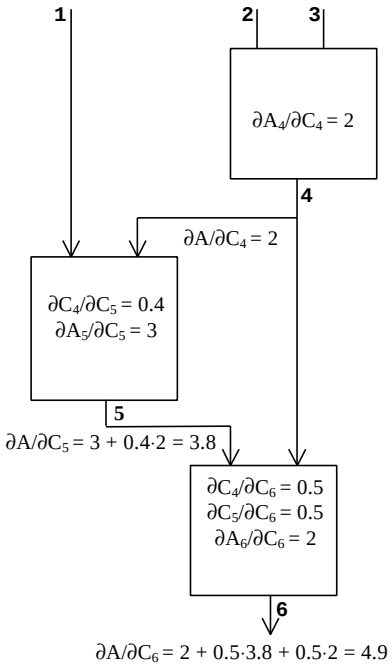

Figure 1: Example for sensitivity calculation

2.4 Implications to Placement

The global area sensitivities $\partial A/\partial C$ derived in Section 2.1 and 2.2 have a very important property. Since they do not depend on any geometrical circuit data like wire lengths or gate sizes, but only on cell parameters that are known for a specific library the sensitivity factors can be calculated once for each gate after technology mapping and do not need to be updated any time during the placement process.

The use of these sensitivity factors allows us to very quickly compute the exact global effect of changing the length of an interconnect wire.

This is possible because the critical path never changes under the constant delay model. When using a conventional constant area model, a net connected to a cell beeing moved might or might not become critical later. In the worst case a full timing analysis is necessary after each cell move to accurately update the circuit delay. Timing driven placement under conventional timing models therefore either employs heuristics to estimate the effects of a cell move or tries to optimize timing indirectly by minimizing secondary criteria such as wire length.

In contrast, under the constant delay model the circuit area can be calculated as a weighted sum of wire lengths. This enables the placer to directly optimize the primary optimization objective. Our experiments show that this added accuracy can significantly improve the placement results.

Care must be taken that the matrix inversion involved in computing the area sensitivities for sequential circuits does not dominate the runtime of the whole placement proocess. It is therefore necessary to use advanced inversion algorithms that exploit the sparse nature of matrix *D*.

3 Experimental Tool Flow

For our experiments we modified our existing timing driven tool flow from [9] to optionally use the constant delay model. The resulting tool flows are shown in Figure 2.

Technology mapping is performed on the basis of a hypothetical cell generator for series-parallel CMOS cells. [8]

Technology mapping under the constant delay model is the problem of covering the technology independent netlist with constant delay cells that achieve the desired target delay using minimum expected area. This involves the selection of a gate cover but also delay budgeting: The circuit delay must be distributed between the cells on each path in a way that minimizes the total circuit area.

In terms of the theory of logical effort [6] the goal of the optimization is to assign larger delays to gates with a larger logical effort and a larger branching effort. Of course, the branching effort depends on the wire length and cannot be known before placement. Therefore the

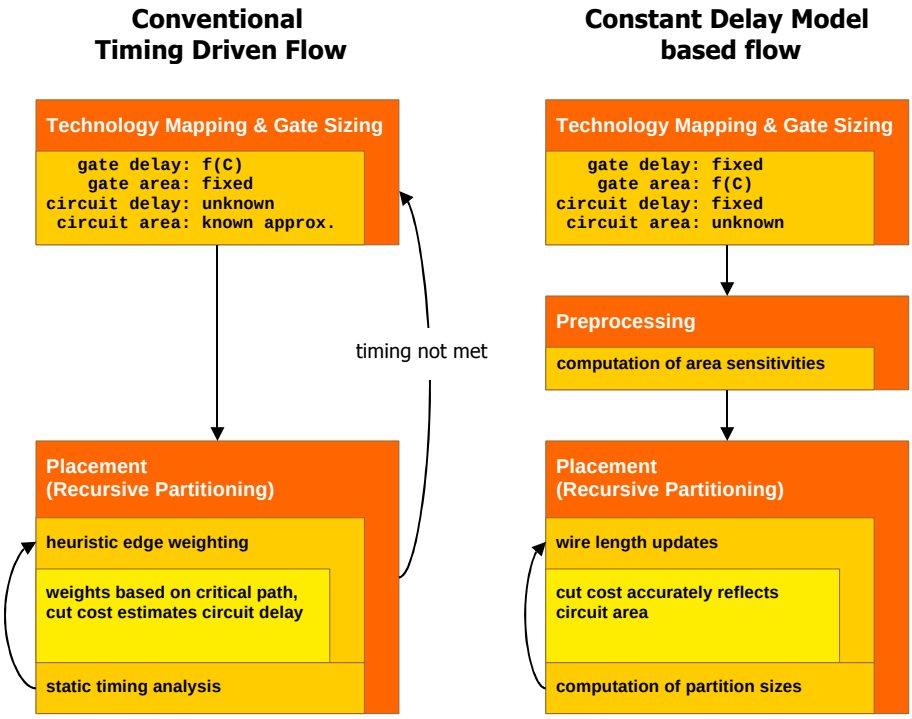

Figure 2: The toolflows used in the experiments

mapper uses rough estimates for the wirelength based on the net fanout. Our tool flow allows to update these estimates and to perform remapping after each placement step.

The mapping algorithm uses an approach similar to [7] but does not perform refactoring of the circuit. A general discussion of technology mapping under the constant delay model can be found in [10]. Our mapper also performs pre-placement buffer insertion and presents an estimated area delay tradeoff curve to the designer who selects the desired design point.

For the conventional tool flow the mapper also performs gate sizing. For the constant delay model the mapper passes the gate parameters to the placer. These parameters are used to compute the global area sensitivities of all gates according to Equation (6).

The placer uses the same FM-based [1] recursive bipartitioning algorithm for both flows. Only the net cost computation has been modified. In both models the cost of a net is based on the change in wiring capacitance caused by changing the length of the wire when the net joins the cut.

For the conventional model this change in wiring capacitance is multiplied by a criticality factor based on a static timing analysis that is performed once for each recursion level. The timing analysis is too time consuming to be performed more often and the criticality factors might therefore become inaccurate during the partitioning process.

For the constant delay model the capacitance change is multiplied by the global area sensitivity that was computed beforehand. As shown above the area sensitivities remain valid throughout the placement process.

For the wire length calculation nets are assumed to be connected to the center of cells. Cells are assumed to be moveable within their region and the wire length used for the cost computation is half the perimeter of the smallest rectangle that overlaps all regions that contain cells connected to the wire.

Region sizes were only updated once for each partitioning level. This can make the wire length calculation inaccurate for the constant delay model if the area change caused by the cell moves distributes very unevenly among the regions. However, in our experiments this turned out to be a minor phenomenon. If necessary, this problem could be overcome by calculating area sensitivities for each region rather than for the global area.

This however would have an impact on runtime. For each of the O(log n) partitioning levels each cell in the circuit is moved once. If only the global area is updated the time complexity of the placer is O(n log n). On the other hand there are 2^l regions at partitioning level l. If each region is updated for each cell move the total number of area updates becomes $n \sum_{l<\log n} 2^l = \Theta(n^2)$.

Circuit	Target delay	Conventional		Conventional iterated		Constant Delay no remapping		Constant Delay remapping	
		Area	Delay	Area	Delay	Area	Delay	Area	Delay
C432	1.80	5.00	2.70	6.34	1.81	5.87	1.81	5.57	1.78
C499	1.64	12.0	2.28	17.5	1.66	15.0	1.68	15.0	1.66
C880	1.50	9.38	2.41	12.8	1.50	10.3	1.48	10.3	1.49
C1355	2.10	14.9	3.24	20.9	2.08	19.5	2.09	17.5	2.14
C2670	1.20	18.4	1.99	26.7	1.21	22.1	1.26	20.9	1.28
C3540	2.73	30.9	4.60	49.2	2.74	36.2	2.74	35.9	2.75
C5315	3.00	30.0	4.44	70.2	3.03	56.4	3.03	56.4	3.08
C6288	1.18	71,5	1.68	83.0	1.18	82.8	1.17	82.8	1.16
C7552	2.60	64.8	4.33	89.0	2.56	75.9	2.59	75.2	2.57
				375,6		324,1		319,6	
				118%		101%		100%	

Table 1: Experimental results

4 Experimental Results

To examine the benefit of our approach a comparison of three different design flows is of interest. As explained in Section 3 all three flows are modifications of our previous work [9].

1. a conventional design flow consisting of standard cell mapping followed by timing-driven partitioning based placement.
2. a single constant delay mapping followed by partitioning using area sensitivities.
3. as in (2), but with additional technology mapping after each partitioning level.

In all cases we assumed continuous gate sizing by a cell layout generator. Our current implementation only includes the computation of area sensitivities for combinational circuits.

The results of our experiments are shown in Table 1. Delay values are expressed in nanoseconds, area values are expressed in 10^3 square micron. Column 1 and 2 contain the name of the circuit and the specified target delay. Columns 3 and 4 show the area and delay results for a conventional design flow. It is easy to see that the finally realized delay is far away from the initially specified target delay because the cell sizes had to be chosen without placement information. In most cases the larger delay results in a smaller area, but overall the results are unpredictable.

In order to compare the circuit areas of the conventional flow to the constant delay model based flows it must be forced to meet the target delay from column 2. Multiple iterations over the whole process trying different target delays and pre-placement net length predictions were necessary. The best achieved results for the conventional flow are shown in columns 5 and 6.

Columns 7 and 8 contain the results for a single constant delay mapping step followed by placement. As expected from theory predefined target delay constraints have been met almost exactly.

The results clearly show that the improved guidance of the partitioning process as proposed in this paper leads to a significantly smaller circuit area when compared to the conventional approach shown in column 5.

In our implementation even repeated remapping obtained almost no additional improvement, as shown in columns 9 and 10. For some of the benchmark circuits the netlist changes considerably during remapping, but apparently these changes have not much effect on the area. This shows that the placer chooses placements that are well optimized for the initial mapping and it can not improve the results by adapting the mapping to the placement. In this sense, our flow is very robust with respect to the initial gate covering.

5 Conclusion

We show that under the constant delay model the circuit area is a weighted sum of the interconnect capacitances.

The weights remain valid throughout the placement process and do not need to be updated as long as the circuit structure is not modified. We presented methods to calculate these weights for combinational and sequential circuits.

The presented results can be applied to a wide range of optimization strategies such as simulated annealing and recursive partitioning and adds the capability to directly optimize circuit area using constant time cost updates without resorting to heuristics based on secondary optimization criteria.

Our experimental design flow based on bipartitioning and the constant delay model does not only match the target delay without iteration but also brings about significant savings in area when compared to a conventional design flow. Our method fits well into a standard bipartitioning framework because only the net weight calculation needs to be adapted.

References

[1] Fiduccia, C and M., Mattheyses, R. M.: "A linear-time heuristic for improving network partitions", Proc. 19th Design Automation Conference, 1982

[2] Fishburn, J. and Dunlop, A.: "TILOS: A polynomial programming approach to transistor sizing", Proc. ICCAD-85, 1985

[3] Dunlop, A.; Kernighan, B.: "A Procedure for Placement of Standard Cell VLSI Circuits", IEEE Transactions on CAD, vol. 4, no. 1, 1985

[4] K. Chaudhary, and M. Pedram: "A nearly optimal algorithm for Technology Mapping minimizing Area under Delay Constraints", Proc. 29th Design Automation Conference, 1992

[5] W. Swartz, C. Sechen: "Timing Driven Placement for Large Standard Cell Circuits", Proc. of the 32nd Design Automation Conference ,1995

[6] I. Sutherland, R. Sproull: "The theory of logical effort: designing for speed on the back of an envelope", Proc. of the 6th MIT conference: Advanced Research in VLSI, 1990

[7] D.-J. Jongeneel et al.: "Area and Search Space Control for Technology Mapping", Proc. 37th Design Automation Conference , 2000

[8] M. Lefebvre, D. Marple and C. Sechen: "The Future of Custom Cell Generation in Physical Synthesis", Proc. 34th Design Automation Conference, 1997.

[9] I. Neumann, D. Stoffel, H. Hartje and W. Kunz: "Cell Replication and Redundancy Elimination During Placement for Cycle Time Optimization", Proc. of the ACM/IEEE Intl. Conference on Computer-Aided Design (ICCAD), 1999.

[10] J. Grodstein, E. Lehman, H. Harkness, B. Grundmann and Y. Watanabe: “A Delay Model for Logic Synthesis of Continously-Sized Networks”, Proc. of the ACM/IEEE Intl. Conference on Computer Aided Design (ICCAD), 1995