Global Stacking for Analog Circuits

Bogdan G. Arsintescu
Delft University of Technology
Mekelweg 4, 2628CD Delft, The Netherlands
Bogdan@CaS.ET.TUDelft.nl

Sorin A. Spânoche
"POLITEHNICA" University of Bucharest
Bvd. I.Maniu 1-3, 77202 Bucharest , Romania
Sorin@messnet.pub.ro

Abstract

A flexible and efficient method for analog circuit partitioning and transistor stacking is presented. The method is based on a novel algorithm dealing with analog specific constraints and on a set of heuristics for stack generation using a pattern database. An enhanced set of stacks is obtained with respect to placement constraints. Experimental results show the effectiveness of the methods described.

1. Introduction

In the last decade, some attempts have been made to develop automation for analog circuit layout design [1, 2, 3, 4]. Since the constraints for analog and digital design are different, the techniques for digital circuits cannot be easily ported to analog design. Analog physical design has to deal with special requirements for matching, symmetry, parasitics and for the variety of transistor sizes. The layout objectives in analog design target layout symmetry and device matching. Typical techniques in analog layout are large device folding, interdigitated structures for symmetrical pairs, geometry sharing and device chaining for neigbouring devices (see Figure 1). All these methods are referred to in literature as MOS transistor stacking. In [5] a design style is proposed for *fully-stacked* layout of analog circuits. That is, all the transistors in a circuit are stacked and a layout module is generated for each stack. The goal of this style is to decrease stray capacitances and to improve the layout of analog circuits.

In this paper we propose a new automated technique for transistor stacking in analog MOS circuits. Our aim is to obtain an enhanced set of stacks for placement and routing. Since we do not have placement and routing information at this stage of the design, we will generate an enhanced list of stacks including trade-offs of the same stack. Among all these stacks we can choose a suitable set according to layout specific constraints.

In [1], an algorithm for the automatic generation of full-stacked layouts in analog CMOS circuits is described. In that work, circuit partitioning in stacks is made on a "same bulk" criterion, i.e. with the same substrate potential. Additional constraints (symmetrical pairs, matching groups) should be explicitly enforced by the user. Firstly, circuit partitioning is done and transistors are split into segments. The algorithm of [6] is used for abutted stack generation, i.e. optimal transistor chaining. A single set of stacks is generated, claimed optimum with respect to a sum of critical parasitics and device area minimization. No placement constraints are considered. The method described in [1] is based on stray capacitance minimization. Therefore it will produce stacks with a big number of transistors (long stacks). The method will fail to generate useful stacks when the sizes of the transistor groups are relative prime numbers with respect to grid size. This may result in tall stacks (equal with the size of a transistor not folded). Both long and tall stacks will be difficult to handle by the placer.

Our stacking method attempts to resolve all the shortcomings mentioned above. We propose a flexible method for generating an enhanced set of stacks, with form factor (W/L) trade-offs for every stack. By allowing small device size variation, we can avoid generation of tall stacks. Based on the set of stacks obtained, better placements are expected.

We propose an automated method for grouping symmetrical pairs and current mirrors. Transistors are split into parallel transistors (called segments) and arranged in a stack according to predefined patterns for all groups of transistors in a circuit. This initial list of stacks is enlarged by concatenation or by merging stacks in order to create all the stack candidates needed for a good placement. We allow device size variation during stacking, within the limits of circuit specification. By allowing size variation, our approach solves the problem of transistor stacking irrespective of initial sizes, while preserving the critical circuit parameters.

Very wide transistors can generate a poor layout, even when stacked. To avoid this, our program can split one transistor in two different stacks, adding more freedom for the placer. This feature is also used for symmetrical pairs, where centroidal twin stacks can be made. In a centroidal

EURO-DAC '96 with EURO-VHDL '96
0-89791-848-7/96 $4.00

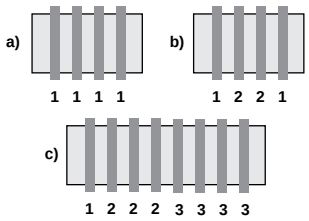

Figure 1. Examples of Stacks. (numbers indicate transistor gate connections)

pair of stacks, each of the twin stacks contains half of the stack segments (see figure 3.b). These structures are useful in small offset applications.

The full list of stacks is generated together with several cover sets. They can serve to find a globally optimum set of stacks according to placement and global routing constraints.

This paper is organized as follows. Section 2 is an overview of the global optimal stack generation method. In Section 3 we describe the algorithm used for circuit partitioning. The stack generation method is explained in Section 4 and 5, for initial stacks and for the entire list of stacks respectively. Experimental results are shown in Section 6, and conclusions are drawn in Section 7. The Appendix lists the definition of several terms used in this paper.

2. Global stacks generation

The generation of MOS Transistors (MOSTs) stack is achieved in four steps, as follows:

1. In the first step, the circuit is partitioned by finding all the symmetrical pairs and matching groups of transistors. The remainder MOSTs are single transistors.
2. An initial set of stacks is subsequently generated by using the elements of the partition, the symmetry information and the compatibility between transistors.
3. The complete set of stacks is then generated by combining each existing stack pair, provided they have compatible sizes and the two stacks do not contain the same MOSTs.
4. Finally, several stack covers for the circuit that can lead to a good layout are found.

For Step 1 we developed an algorithm that finds all symmetrical pairs and current mirrors, based on circuit topology information in a bipartite adjacency graph. One set of vertices is the set of circuit nets and the other one is the set of MOSTs terminals. An edge between two vertices exists if the corresponding net of the start vertex is connected to the corresponding MOST terminal vertex. The adjacency graph is obtained by parsing a net-list file with transistor dimensions and technology information. This graph structure helps extracting circuit topology information. Based on the results of the algorithm, we create a partition of the circuit containing transistor pairs and matching groups identified by the algorithm and single transistors not included in the previous categories.

All the groups in a partition are stacked separately in Step 2. For each group we use a set of appropriate stack models, taken from a user-definable database that contains the generally used patterns (i.e. stack models) in analog layout design.

During Step 3 the stacks are merged starting from this list of initial stacks. We enhance the initial set of stacks with stacks containing three or more MOSTs. These stacks decrease parasitic capacitances of the nets abutted in the stack and improve matching between devices. The resulting stack is evaluated in terms of circuit performance, prior to insertion in the list.

A circuit evaluation is done in order to test whether the circuit updated with the generated stack(s) dimensions and electrical characteristics can lead to a circuit that performs according to the design specifications. Due to the fact that MOSTs sizes are changed during the new algorithm, parasitics bounds as in [7] are not applicable. An automated approach for generating the circuit equations (symbolic simulation) as in [8] or a direct electrical simulation can be used. We however feel that this circuit dependent analysis has to be user definable, i.e. based on the specific circuit equations, for reasons of speed and because a user rarely lays out a circuit that she cannot model.

An optimum circuit cover by stacks can be found during Step 4, subject to a cost function as described in [1]. The quality of the resulting cover may not be satisfactory during placement and routing. Our solution is to find several covers using a greedy algorithm. Firstly we apply a circuit evaluation cost function to the initial circuit and the sized stack. A new stack is selected at each step if it minimizes a cost function of circuit evaluation together with the previously selected stacks. The sizes of circuit elements contained in it are updated in the cost function calculation and a new greedy step is performed. The greedy algorithm stops when cover conditions are met. Starting from this cover and the global stack list, further optimization will take place during the placement phase.

3. Symmetrical and matching groups

An algorithm capable of recognizing symmetry in a fully symmetrical electronic network graph was proposed in [9]. Analog circuit design seldom leads to fully symmetrical circuits (i.e. a perfect mirrored circuit structure relative to an axis). We propose an algorithm that finds all symmetrical groups with respect to topology patterns specific for analog circuitry in general circuit structures.

```
Algorithm: SYM
  INPUT: circuit bipartite graph
  FOR A = every net in the circuit graph
    FOR B = all MOSTs connected to A
      Mirror-Group = WBM(A, B, Cost-Mirror)
      Symmetrical-Group = WBM(A, B, Cost-Symmetry)
      IF (SymmetricalGroup ≠ ∅)
        AugmentPath(Symmetrical-Group)
      END-IF
      SAVE Mirror-Group, Symmetrical-Group
    END-FOR
  END-FOR
  OUTPUT Mirror-Group(s) and Symmetrical-Group(s)
END_main

AugmentPath(S)
  FOR all A = the net pair connected to S
    FOR B = all MOSTs connected to A
      Symmetrical-Group = WBM(A, B, Cost-Symmetry)
      IF (S ≠ ∅)
        Augment(S)
      END-IF
      RETURN Symmetrical-Group
    END-FOR
  END-FOR
END_augment
```

Figure 2. The algorithm flow

The following patterns are used:

- **simple current mirror pattern:** a set of MOSTs with the gates connected to the same net and the sources connected to the same net accordingly.
- **symmetrical pair:** a pair of identical MOSTs connected with the same terminal to the same net, or to corresponding terminals of another topological symmetric pair.

The algorithm is described in Figure 2. We use Weighted Bipartite Matching (WBM) algorithm in a bipartite adjacency graph structure. The bipartite sets of vertices correspond to the list of nets and to the list of MOSTs terminals respectively. An edge connects a vertex in the first set to a vertex in the latter if an electrical connection exists between the corresponding net and MOST terminal. The WBM algorithm will return a nonempty set of MOST *only* if a pattern is found. The heuristic functions $CostMirror$ and $Cost\text{-}Symmetry$ determine the edge weights according to the patterns definitions. WBM results of cost zero are not interesting, since they do not imply a pattern match. The desired MOSTs groups are derived from the WBM results.

The $AugmentPath$ function will recursively search for new symmetrical pairs connected at the same terminal of an already found symmetrical pair. The depth of the recursive procedure is limited to $n/2$, where n is the number of MOSTs in the circuit in case the circuit is completely symmetrical. Our algorithm avoids duplicating symmetrical structures. The maximum number of $AugmentPath$ calls is therefore $n/2$. A "clean–up" procedure is necessary to remove the MOST's included in both symmetry groups and matching groups, by removing them from the latter type of groups. In this way symmetrical pairs are preserved as much as possible.

4. Initial set of stacks

The heuristics used in generation of the initial stacks narrow the big number of possible solutions down to a smaller number of stacks useful for good layout [1]. The maximum number of transistors stacked at this level is two (symmetrical pairs). The number of patterns in database will, hence, be small.

We build the initial set of stacks for each element of a partition according to the following heuristic rules:

- A stack can have no more than 5 segments per MOST, except for large form factor (W/L hereafter). Stacks with more segments per MOST will be very long, considering also that they have to be linked with other stacks.
- the segment size is bounded between a minimum size and a maximum size. At least one stack is made if all attempts are out of bounds. These bounds help to preserve the form factor of resulting stacks.
- The single MOSTs are split in simple folded stacks (see Figure 3.c). If the width of a single MOST exceeds a certain limit, it can be split in two parallel connected transistors, stacked separately (see Figure 3.d). This kind of structures can be used to complete a L-shaped layout to a rectangle.
- The symmetrical pairs can have an interdigitated structure, i.e. single (see Figure 3.a) or twin (see Figure 3.b) or a normal folded structure, separated by an insulating dummy segment. The dummy segment is either a MOST biased in its off state or a space between diffusions. The choice depends on the design rules.
- The folded structure with dummy segment is allowed only if the MOSTs to be stacked have no direct connection between sources or drains.
- The mirror groups are stacked transistor by transistor and the resulting stacks are merged if possible.
- Each stack is tested by the circuit evaluator prior to list insertion.

For each element of the partition (transistor or group of transistors) we search in the database for a class of appropriate patterns. Stacks are generated for all these patterns and according to the above mentioned rules.

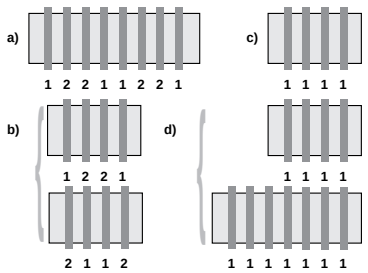

Figure 3. Examples of Stack Patterns

At least one stack is generated for each element. That is, if the size of the MOST is smaller than the lower bound, a stack with one segment per MOST is generated. If the MOST size exceeds the upper bound, then a stack with 5 segments per MOST is generated. A special case are single, large W/L MOSTs. These MOSTs will be split in two stacks.

The pattern database contains a collection of patterns, selected by analog design experts. The user can add or discard patterns to/from the database. Examples of patterns used are given in figure 3. We use the Euler Path algorithm and the heuristics mentioned in the beginning of this section in building the database. We add dummy edges in the multi-graph corresponding to the electrical structure of the pattern until it becomes an Euler graph. Dummy edges will be dummy transistor segments in the resulting stack. Any Euler path in this graph is equivalent with an electrically correct stack. The selection of proper solutions among all the possible ones is made at the end of the algorithm.

Other solutions were considered for building initial stacks. The algorithm described in [1] generates all the cliques. Another possibility was to find all Euler paths in a multi-graph [10] G(V, E) where V is the set of nets and E are the transistor segments. This algorithm proved to be faster than using cliques. The chosen solutions were not different from the patterns from the database. None of the trade-offs that were presented in this paragraph can build twin stacks for full centroidal design. They cannot have the freedom to split a very wide transistor in two and stack the two parts separately. The algorithmic approach proved to be slower and less flexible that the pattern database search.

As stated before, our approach does not preserve exact initial transistor sizes. When splitting a MOST, the size of the segments have to fit the technology grid. Considering only an integer grid, we change both sizes of the element in this step as follows:

$$W_{segment} = RoundInteger(\frac{W_{MOS}}{n})$$

$$L_{segment} = RoundInteger(L_{MOS} * \frac{W_{segment} * n}{W_{MOS}})$$

where W, L are the transistor dimensions of the segment or MOST as the indices indicate, n is the number of segments per MOST and $RoundInteger(a)$ is a function returning the integer closest to a. In case of an integer grid, for $n \leq 5$ the size (length and width) variation is at most 8% (at 5 segments per stack). This variation should be acceptable with respect to specifications of correctly designed circuits. The stacks are evaluated in terms of circuit performance. In case size variation is unacceptable, they are rejected.

This procedure preserves W/L almost invariable. Small variation in the form factor can occur because the sizes are rounded to an integer value. The theoretical variation is maximally 4%. In practice, an average variation of 1.5% is observed.

By allowing size variation we increase the flexibility of the stack generation. Without size variation, the solution set obtained by stacking will be small and uninteresting for placement, unless "friendly" sizes are imposed during the circuit sizing phase. Our approach can build correct stacks irrespective of initial transistor sizes, without changing the critical circuit parameters.

5. Extended set of stacks

Starting from the initial stack list we try to merge pairs of existing stacks using the following set of rules:

1. Both stacks must have the same type and the same bulk connection.
2. None of the two stacks have segments from the same MOST.
3. The widths difference of the stacks in a pair is acceptable.

For the first rule we check whether we can place both stacks on the same diffusion island. The second rule prevents overlapping, i.e. the same MOST should not appear twice in the same stack.

A pair of twin stacks existing in the initial stack list are referred to as native twin stacks. Native twin stacks are excepted from the second rule, because a combination between them will produce an already existing stack. Once one stack in a twin pair is merged with another stack it becomes eligible for merging with its twin or its descendants. Two dimensional centroidal structures can be obtained only by two stacks, each of them containing one or both native twin stacks.

Rule 3 checks if the relative difference in size (i.e. segment width) between the two stacks is acceptable. A relative size variation of 10% can generate stacks that are usually accepted by the circuit evaluator.

The new stacks are generated in 3 ways: (1) by concatenation, (2) by insertion or (3) by re-generation. Concatenation is applied in case two external segments of the stacks are connected to the same net (figure 4.a) and by inserting

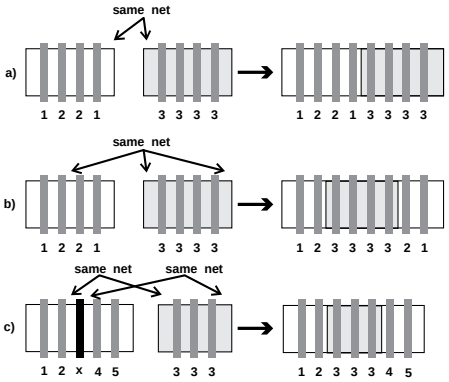

Figure 4. Examples of stack merging

a dummy segment between the two stacks. Insertion of a stack in another is used in case the dummy segment in one stack can be replaced by the second stack. Insertion can be also used in case the connecting net is in internal position in one stack and in external position at both ends for the other stack (figure 4.b).

Real life cases are generally resolved by the first and second method. Regeneration uses the Euler Path algorithm for the trail generated by the graphs of the two stacks.

We modify the sizes of the segments in the new stack choosing a suitable combination of the following methods:

- keep each segment at either $L = constant$ or at $W/L = constant$ and
- modify the first, the second or both input stack sizes.

The newly generated stack is optimistically evaluated by the circuit evaluator. That is, the cost function evaluates the circuit with the transistors obtained from the new stacks. All other transistors have their original sizes. If the specifications hold, then the new stack is accepted, and inserted in the list.

This procedure is iterated several times, until no new stack is generated or a maximal iteration count is reached. All our tests were convergent after at most three steps (i.e. no new stack was generated).

6. Results

Our new method of transistor stacking was implemented in the Atlast (Analog Transistor LAyout STacking) program using the "C" programming language. The stack pattern database was generated automatically and then improved by analog designers. The circuit evaluator, which depends on circuit topology, is described by the user using a "C" library targeted for circuit modelling. This approach will probably be changed in the future to an equation evaluator, even if the current solution leads to short run time.

Atlast program was run on several medium size circuits

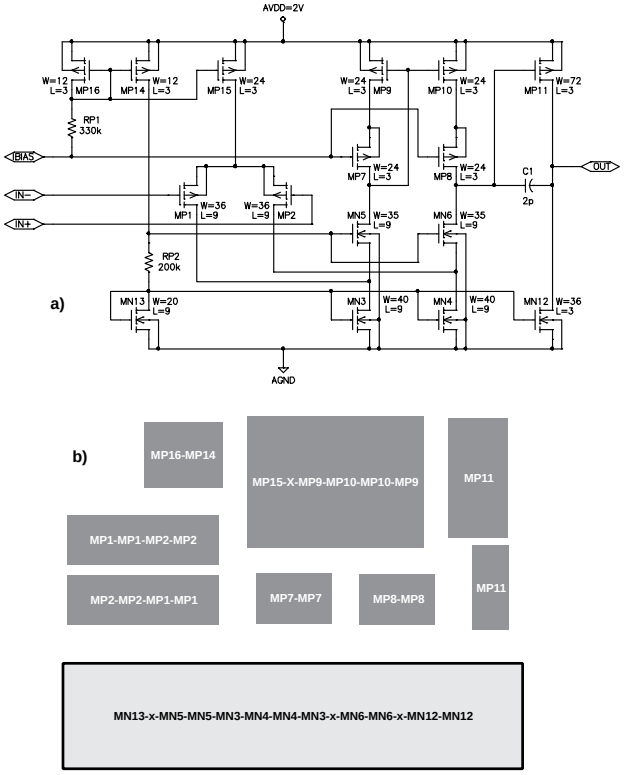

Figure 5. Stacking Results for a Two Stage OPAMP

(10 . . . 20 MOSTs) and the resulting solutions were placed using heavy stacking and centroidal patterns. The solutions were tested, when possible, against manual layout. One top covering stack set was identical to the manual solution in all cases.

Figure 5 shows some results generated by Atlast for a 16 MOSTs, folded, two stage OPAMP (5.a) from a low voltage circuit. In the cover shown in figure 5.b , one can notice a stack that contains all NMOS transistors and two dummies. This was possible by transistor size modification during the run even if the greatest common divisor of the transistor widths is one. The resulting stack width is $20\mu m$.

The input pair MP1, MP2 can be placed using a centroidal scheme using the two stacks at middle-left. Transistors MP7 and MP8 were laid out separately due to their different bulk connection. The MP11 MOST was split in two stacks (right-top) so that, together with the missing components: C1, RP1 and RP2, it can offer higher flexibility in the placement phase. The pair of stacks for MP1-MP2 MOSTs, the pair for MP11 and the NMOS stack can be now generated with our method. The previous approach of [1] can not handle such structures due to limitations in terms of initial MOSTs sizes and stacking strategy.

Table 1 summarizes some of the electrical characteristics

Electrical Parameter	Pessimistic initial simulation (SPICE)	Optimistic initial estimation (Atlast)	Optimistic final estimation (Atlast)	Final simulation (SPICE)
Gain-Bandwidth Product$[MHz]$	1.30	1.50	1.41	1.38
Phase Margin $[deg]$	62	75	68	67
Noise Corner Frequency$[kHz]$	10.1	6.1	7.5	8.2
Slew Rate $[V/\mu - second]$	1.3	1.5	1.4	1.3

Table 1. Comparative Electrical Characteristics

of the circuit as simulated with SPICE and estimated during the Atlast run. The estimator used a code based on hand design and modelling procedures.

The initial, pessimistic simulation was done using maximal source-bulk and drain-bulk capacitances. This is a usual, pre-layout procedure in the industry. Optimistic estimation takes into account minimal or real node capacitances. One can see that there is room for performance degrading during the routing phase before the pessimistic simulation results are reached.

7. Conclusions

An original approach for generating stacks of MOS transistors for analog circuit layout design was presented. Using the new, heuristics based method it is possible to generate stacks similar to the manual analog layout. These contain 2D centroidal structures, folded devices, chained devices and large devices that are split in two stacks.

The method is based on a symmetry finding algorithm, pattern based stack generation, stack merging and cover finding. The electrical circuit characteristics are checked at each new stack using an optimistic evaluator. A circuit example has been reported showing the flexibility of the results that can be obtained with this method.

The authors would like to thank (a) their Internet providers that enabled the virtual work environment and (b) their friends - analog designers. This research has been partially supported by STW contract no. EEL 33.3157 at TUDelft and grant no. TB-63/5001C by Romanian Ministry of Education and Science.

Appendix

The following are definitions of several terms used in this paper.

Definition 1 (segment) *is a layout entity corresponding to a part of a MOS transistor layout. A transistor is split in some transistor segments connected in parallel. If a single MOST is split in n segments (hereafter Sg), then we have the following rules:*

$L_{Sg_i} = L_{MOS}\ and\ \sum_{i=1}^{n} W_{Sg_i} = W_{MOS}$

where W_{Sg_i} is the width of segment i and L is the length of the MOST or of the segment, as given by the indices.

Definition 2 (stack) *a chain of segments which partially builds an electrically correct layout for a set of MOS transistors in the circuit. (see Figure 1)*

Definition 3 (cover) *a set of stacks $\mathcal{S}$ of a given circuit with the set $\mathcal{M}$ of MOST, each $M \in \mathcal{M}$ split in segments Sg_M is a cover if:*

$\forall Sg_{M_a} \in S_1, Sg_{M_b} \in S_2\ [\ S_1 \neq S_2 \Rightarrow M_a \neq M_b\ ]$ *: non-overlapping condition and*

$\forall M \in \mathcal{M}, \exists S \in \mathcal{S}\ [\ \forall Sg_M \in M \Rightarrow Sg_M \subseteq S\ ]$ *: injectivity condition.*

References

[1] E. Malavasi and D. Pandini. Optimum CMOS stack generation with analog constraints. *IEEE Trans. on CAD*, 14(1):107–120, Jan. 1995.

[2] S. S.W. Mehranfar. A technology-independent approach to custom analog cell generation. *IEEE J. Solid-State Circ.*, 26(3):386–393, Mar. 1991.

[3] E. Rijmentants, J. Litsios, T. Schwarz, and M. Degrauwe. ILAC: An automated layout tool for analog CMOS circuits. *IEEE J.of SSC*, 24(2):417–425, 1989.

[4] J. Cohn, D. Garrod, R. Rutenbar, and L. Carley. *Analog Device-level Layout Automation.* Kluwer, 1994.

[5] U. Gatti, F. Maloberti, and V. Liberali. Full stacked layout of analogue cells. In *Proceedings ISCAS*, pages 1123–1126, 1989.

[6] S. Wimer, R. Pinter, and J. Feldman. Optimal chaining of mos transistors in a functional cell. *IEEE Trans. on CAD*, 6(5):795–801, May 1987.

[7] H. Chang et al. A top-down, constraint-driven design methodology for analog integrated circuits. In *IEEE Proceedings at Custom Integrated Circuits Conference*, pages 841–846, 1992.

[8] K. Lampaert, G. Gielen, and W. Sansen. A performance-driven placement tool for analog integrated circuits. *IEEE J. of SSC*, 30(7):773–780, 1995.

[9] M. Kole and O. Herman. Modeling symmetry in analog electronic circuits. In *Proceedings ISCAS*, pages 315–318, 1994.

[10] A. Basaran and R. Rutenbar. An $o(n)$ algorithm for transistor stacking with performance constraints. In *ACM/SIGDA Physical Design Workshop*, pages 262–267, 1996.