Tabu Search Based Circuit Optimization

Sadiq M. Sait, Habib Youssef, and Munir M. Zahra

Department of Computer Engineering
King Fahd University of Petroleum and Minerals
KFUPM Box 673, Dhahran-31261, Saudi Arabia
Tel: 966(3)-860-2110/2217 *e*-mail: {sadiq,youssef}@ccse.kfupm.edu.sa

Abstract

In this paper we address the problem of optimizing mixed CMOS/BiCMOS circuits. The problem is formulated as a constrained combinatorial optimization problem and solved using an tabu search algorithm. Only gates on the critical sensitizable paths are considered for optimization. Such a strategy leads to sizable circuit speed improvement with minimum increase in the overall circuit capacitance. Compared to earlier approaches, the presented technique produces circuits with remarkable increase in speed (greater than 20%) for very small increase in overall circuit capacitance (less than 3%).

Keywords: Tabu Search, Circuit Optimization, Search Algorithms, CMOS/BiCMOS, Mixed Technologies, Critical Path, False Path.

1 Introduction

Popularity of CMOS technology is due to its low DC power dissipation and high package density. The demand for superior performance motivated research and development that lead to the emergence of BiCMOS technology. BiCMOS is a combination of CMOS and Bipolar technologies, with advantages of both, high speed and high driving capabilities of Bipolar, as well as the low area and low power consumption of CMOS.

VLSI designs are evaluated with respect to three main performance criteria: speed, area, and power consumption. As these criteria are conflicting designers usually seek to optimize one criteria, namely speed, while satisfying specific constraints/requirements on area and power consumption.

One of the optimization techniques applied at the circuit level is the selection of logic blocks of the VLSI circuit, in terms of speed and area. For example, for standard cell designs, the optimization can be performed through a careful selection of different implementations of a block in the same technology. These alternative implementations vary in area, driving capabilities, intrinsic delay, and capacitive loading [11]. Another optimization strategy is to follow a mixed technology design approach. One possible choice is to mix CMOS/BiCMOS technologies. In terms of manufacturing process, this is feasible since the CMOS process is part of BiCMOS process. The CMOS-based BiCMOS process is a CMOS baseline process to which bipolar transistors are added. So, for a mixed design circuit, initially all cells are exposed to CMOS process. Then bipolar transistors are added to only those cells that are selected to be BiCMOS.

In this paper we discuss the problem of optimizing mixed CMOS/BiCMOS circuits in terms of delay, power and area. Although the scope of the work is directed to CMOS and BiCMOS technologies, other technologies can be included taking into consideration the feasibility and practicality of mixing these technologies.

The basic idea is as follows. Given a circuit consisting of only CMOS cells, some of those cells are selected and replaced by their equivalent BiCMOS cells in such a way that the entire delay of the circuit is decreased with a minimum increase in power and area.

In [1], the above approach for optimizing standard cells circuits is used. The technique aims at improving circuit performance by making for each gate, a choice between CMOS or BiCMOS cells depending only on their load capacitance. The authors reported noticeable speed improvement on all the test circuits used. However, in their implementation, no constraints on power dissipation were placed, and the number of BiCMOS gates was high. The reported approach suffers from several problems, namely:

1. All the nodes are considered for optimization.
2. Output nodes are replaced whether they are on time critical paths or not.
3. The approach is local; that is, it performs the optimization on a single node. It does not have a global view of the circuit; hence it is expected to get trapped at a local optimum solution.

The actual delay of a circuit is determined by the delay of its longest *sensitizable path.* A sensitizable path is a path which can be activated by at least one input vector. Those paths which cannot be activated by any input vector are called *false paths.* A path is *critical* if its total delay is greater than a threshold value. Thus, the problem of finding and estimating the delay of critical paths is called the *critical path problem* [5]. For static timing analysis techniques, the circuit is modeled as an directed acyclic graph in which three popular algorithms are used to trace the paths: Depth First Search (DFS) with/without pruning, Breadth First Search (BFS), and PERT-like trace.

In this work we enumerate all sensitizable critical paths according to the α-critical concept. First, all paths with average delay (including estimation of interconnect delays) exceeding an estimated threshold value are enumerated. From amongst these paths, only sensitizable paths are reported.

In the following section we discuss the α-critical approach. Discussion on false path problem is given in Section 3. The circuit optimization problem is formulated as a combinatorial optimization problem in Section 4. Details of application using tabu search are given in Section 5. Experimental results are provided in Section 6.

2 The α-Critical Approach

The total number of paths in a VLSI design grows exponentially with the size of the design. However, usually a small subset of these paths are timing critical. A path π is classified as critical if its total delay, T_π, is very close to its latest required arrival time $LRAT_\pi$. If T_π exceeds $LRAT_\pi$, path π becomes a long path. The path delay consists of two components: the logic delay which is known prior to layout, and the interconnect delay which is unknown. The interconnect capacitance is a key element in the total interconnect delay.

Let $\pi = \{v_1, v_2, \cdots, v_p\}$ be a path in the circuit graph, where v_1 and v_p are the source and sink cells. The total delay on π is given by,

$$T_\pi = \sum_{i=1}^{p-1}(CD_{v_i} + ID_{v_i}) \qquad (1)$$

where, CD_{v_i} is the switching delay of cell v_i and ID_{v_i} is the interconnect delay of the net driven by cell v_i which may be expressed as follows,

$$ID_{v_i} = LF_{v_i} \times C_{v_i} \qquad (2)$$

where, LF_{v_i} is the load factor of the output pin of the driving cell v_i, expressed in units of time per unit of capacitance, and C_{v_i} is the total interconnect capacitance (area + fringe) of the net driven by cell v_i.

The interconnect capacitance C_{v_i} is estimated using data from past designs as follows. The average and standard deviation of net length for different types of nets (2-pin, 3-pin, ...,m-pin) are collected from past designs of similar complexity[1]. These are transformed into interconnect capacitances. Let $\overline{C_{v_i}}$ and s_{v_i} be the estimated expected interconnect capacitance and standard deviation of the net driven by cell v_i. Then, the expected interconnect delay $\overline{ID_{v_i}}$ of net v_i and its corresponding variance $S^2_{v_i}$ are estimated as follows:

$$\overline{ID_{v_i}} = LF_{v_i} \times \overline{C_{v_i}} \quad ; \quad S^2_{v_i} = LF^2_{v_i} \times s^2_{v_i} \qquad (3)$$

[1]this classification helps reduce the sample variance around the mean

Under the assumption of statistical independence between the nets, the expected delay and variance on any path π can be expressed as follows,

$$T_\pi = \sum_{i=1}^{p-1}(CD_{v_i} + \overline{ID_{v_i}}) \quad ; \quad S^2_\pi = \sum_{i=1}^{p-1} S^2_{v_i} \qquad (4)$$

Let $T_{\max}$ be the expected delay of the longest path in the circuit, that is,

$$T_{\max} = \max_{\pi \in \Pi}(T_\pi) \qquad (5)$$

where Π is the set of all paths in the circuit graph G.

Definition: A path π is α-critical iff:

$$T_\pi + \alpha S_\pi \geq T_{\max} \qquad (6)$$

The parameter α is supplied by the user and is interpreted as a *confidence level*. Then, all paths that satisfy the condition of Equation 6 are reported as the α-critical paths.

3 False Path Problem

The presence of false paths has many undesirable effects which include loss of accuracy and waste of optimization effort. False paths exist in a circuit because of several reasons, namely:

1. *Incompatible transitions:* A false path results from the combination of incompatible transitions.
2. *Incorrect signal flow:* Timing verifiers that operate at switch level encounter this problem. Due to the bidirectional nature of MOS transistors, the intended signal flow in structure such as barrel shifter is not always obvious [2].
3. *Logic dependency:* The most explicit source of false paths comes from some logic that depends on the output of other logic [13].

In recent years many techniques have been proposed to detect false paths. The reported techniques rely on various path sensitization criteria which fall into three types: *Static, Dynamic* and *Viable.*

Static Sensitization: These techniques are based on the D-Algorithm which is widely used in testing. One important assumption of the D-Algorithm is that, except of the signal to be propagated, all other signals in the circuit are assumed to have static values throughout the propagation process. This assumption, however, is not always true which will lead to incorrect results. First it will report some paths as false paths while in reality they are not. Second, it can underestimate the sensitizable path length [3, 2, 15].

Dynamic Sensitization: This approach is also based on D-algorithm, but it takes into consideration the stability requirement of the signals. The false paths reported in [6, 13, 8, 4] follow this approach.

Viability: The flaw in the dynamic sensitization condition is the absence of perfect knowledge on gate delays. In such case, both the exact value of any node at any time before the node has settled to a final value, and the time at which a node settles to a final value is problematic. That is, the above mentioned methods use criteria which are non-robust.

McGeer and Brayton [12] developed a technique that computes the longest viable path in combinational circuits. Their techniques are based on two conditions: correctness and robustness. These two requirements are derived form the idea of Boolean difference. The sensitization criterion associates the path with a Boolean expression, which represents the set of input vectors that activate the path. If the associated Boolean expression of the path is found equal to logic 0, the path is claimed to be a false path.

4 Optimization of BiCMOS/CMOS VLSI Designs

The problem of optimizing a mixed technology design can be formulated as an optimization problem and solved using a variety of algorithms. For a mixed CMOS/BiCMOS design, there exist 2^m solutions for a circuit with m gates. Hence, full/brute force design space exploration is infeasible even for designs of moderate sizes.

Recently, several heuristic techniques have been proposed to find optimal and near optimal solutions to a wide variety of classical hard optimization problems. Some of these popular techniques include *simulated annealing, genetic algorithm* and *tabu search.*

Tabu Search is a *metaheuristic* which can be used as an independent search technique or as a higher level heuristic procedure for solving combinatorial optimization problems. It is designed to guide other methods to escape the trap of local optimality. TS operates by incorporating flexible memory functions to forbid transitions (*moves*) between solutions that reinstate certain attributes of past solutions. Attributes that are not permitted to be reinstated are called *tabu*, and are maintained in short-term memory called *tabu list.* After a specified duration they are removed from the list and are free to be inserted again.

For a variety of problems, TS has found solutions superior to the best solution previously obtained by alternative methods. In other cases, it has demonstrated advantages such as ease of implementation, or the ability to handle additional considerations such as constraints not encompassed by an original problem formulation. [7]

4.1 Problem Definition

Given a netlist of CMOS gates, the objective is to find an optimal or near-optimal solution to the problem of replacing some of the CMOS gates with BiCMOS gates such that the overall delay is minimized with minimum increase in the power and area of the circuit. Since the delay of a circuit depends only on the longest sensitizable paths, only gates belonging to those paths are considered in the search space. The reader should note that changing the implementation of a gate instance from CMOS to BiCMOS does not always result in delay improvement. The BiCMOS gate has less delay than the corresponding CMOS gate if its fanout load F_i is greater than a certain threshold C_i [1]. Therefore only those nodes with $F_i > C_i$ should be considered in the search.

Given a circuit with m nodes and K sensitizable paths, we should first extract all nodes i that are included in the sensitizable paths and satisfy the inequality $F_i > C_i$. Let $A = \{1, 2, \cdots, n\}$ be the set of such nodes and let n be the size of this set.

Let d_i^c and d_i^b be the delay of gate i implemented in CMOS and BiCMOS respectively. The circuit delay gain due to changing gate i implementation from CMOS to BiCMOS is

$$g_i = d_i^c - d_i^b \qquad (7)$$

Obviously, for all $i \in A$, $g_i \geq 0$. Let C_i^c and C_i^b be the total capacitance of the circuit when cell i is CMOS and BiCMOS respectively, with all other gates implemented in CMOS.

$$\Delta C_i = C_i^b - C_i^c \qquad (8)$$

We express the changes in power in terms of changes in capacitance. The power of CMOS and BiCMOS gates is proportional to their capacitive load.

The problem is to find a subset S of A such that when the nodes of S are implemented in BiCMOS will lead to maximum reduction in $T_{\max}$, while satisfying a threshold constraint on capacitance, that is,

$$\sum_{i \in S} \Delta C_i \leq C_T \qquad (9)$$

C_T is a given threshold which represents the maximum allowable total capacitance increase.

Let $T_{\max}$ and C_o be the initial operational clock period and the total capacitance of the circuit when all gates are in CMOS. Assume that the optimization algorithm reduces the overall delay by ΔD and increase the overall capacitance by ΔC. Then, the final operational clock and capacitance of the circuit are

$$T_{\text{clock}} = T_{\max} - \Delta D \qquad (10)$$

$$C = C_o - \Delta C \qquad (11)$$

Furthermore, $C \leq C_{\mathrm{T}}$. The circuit optimization problem as defined above is NP-hard.

4.2 Proposed Solution

To identify the gates of set S, we proceed in three steps:

1. Generation of the α-critical paths of the input circuit.
2. Eliminate the false paths.
3. Apply TS algorithm to select the gates of subset S among those covered by the sensitizable critical paths.

Phase I: This phase enumerates all critical paths according to the α-criticality described in Section 2. Path enumeration is achieved via a PERT-like trace of the circuit graph [18].

Phase II: After generating all critical paths for a given circuit, these paths are checked via false path checking procedure to extract the false paths. This step is necessary to speed up the optimization process by minimizing the input population as well as it gives accurate timing. As mentioned earlier, many techniques have been proposed for this purpose. We use the algorithm of [6, 17] with few modifications. The first modification is the use of a more accurate delay model [18]. Other modifications are related to the way of handling the generation of new events in the events propagation phase.

Phase III: This is the optimization phase. The input of this step is a set of the longest sensitizable paths of the given circuit. The process aims to replace some of the CMOS gates by BiCMOS gates in order to optimize the circuit for delay without exceeding a capacitance threshold constraint. The output is a mixed CMOS/BiCMOS circuit with optimum cost. This optimization step is achieved using Tabu Search heuristic.

5 Tabu Search (TS) Algorithm

Tabu search is an iterative procedure that works by making moves from one trial solution to another. An algorithmic description of a simple implementation of the TS is given in Figure 1.

The TS procedure starts from an initial feasible solution s (current solution) in the search space Ω. A neighborhood $\aleph(s)$ is defined for each s. A sample of neighbor solutions $\mathbf{V}^* \subset \aleph(s)$ is generated called *trial* solutions ($n = |\mathbf{V}^*| \ll |\aleph(s)|$), and comprises what is known as the candidate list. From this generated set of trial solutions, the best solution, say $s^* \in \mathbf{V}^*$ is chosen for consideration as the next solution. The move to s^* is considered even if s^* is worse than s, that is, even if $cost(s^*) > cost(s)$.

In order to prevent returning to previously visited solutions a memory or list $\mathbf{T}$, known as *tabu list*, is maintained. Whenever a move is accepted, its attributes are introduced into the tabu list. The purpose is to prevent the reversal of moves for the next $k = |\mathbf{T}|$ iterations because they *might* lead back to a previously visited solution.

In certain situations, it is necessary to overrule the tabu status. This is done with the help of the notion of *aspiration criterion*. Aspiration criterion overrides the tabu status of moves whenever appropriate. One aspiration criterion, also known as *best solution* aspiration criterion (AS_1) overrides the tabu restriction if the move produces a new best solution. In this work we used this and another aspiration criterion, called *aspiration by search direction* (AS_2) [14]. In aspiration by search direction, if an improving move e is made, then the reverse move $\overline{e}$ is accepted if it also is an improving move.

Algorithm: Short-Term Tabu Search

Ω	:	Set of feasible solutions.
s	:	Current solution.
s^*	:	Best admissible solution.
c	:	Objective function.
$\aleph(s)$	:	Neighborhood of $s \in \Omega$.
$\mathbf{V}^*$	:	Sample of neighborhood solutions.
$\mathbf{T}$	:	Tabu list.
$\mathbf{AL}$	:	Aspiration Level.

```
     Begin .
1.   Start with an initial feasible solution $s \in \Omega$.
2.   Initialize tabu lists and aspiration level.
3.   For fixed number of iterations Do
4.       Generate neighbor solutions $\mathbf{V}^* \subset \aleph(s)$.
5.       Find best $s^* \in \mathbf{V}^*$.
6.       If move $s$ to $s^*$ is not in $\mathbf{T}$ Then
7.           Accept move and update best solution.
8.           Update tabu list and aspiration level.
9.           Increment iteration number.
10.      Else
11.          If $c(s^*) < \mathbf{AL}$ Then
12.              Accept move and update best solution.
13.              Update tabu list and aspiration level.
14.              Increment iteration number.
15.          EndIf
16.      EndIf
17.  EndFor
     End.
```

Figure 1: Algorithmic description of short-term TS.

Initial solution: The initial solution can be any feasible solution. It is found that TS may take longer if given a poor initial solution. In our case, the initial solution is a set A of nodes of type CMOS only, which are covered by the sensitizable α-critical paths. As search proceeds, neighbor solutions are generated by swapping CMOS gate by BiCMOS gate or vice versa. The selection of the gate for swap is done randomly or based on a goodness factor.

Tabu list: Formulation of the tabu list is one of the main steps in TS. Since we have only one type of move we used one tabu list. Each entry in the tabu list contains the following information:

- gate number in the path,
- gate type, CMOS or BiCMOS,
- cost associated with this move,
- frequency of the move, and,
- a gain bit (0 or 1): this field is used if *aspiration by search direction* is used as discussed later.

Evaluation function: In order to select the best solution among several candidate solutions generated in each iteration we have to evaluate each solution. The evaluation function is formulated to incorporate all the parameters to be optimized. As mentioned earlier, our objective is to maximize $\sum_{i \in S} g_i$, that minimizes $T_{\max}$, subject to capacitance constraints. The evaluation function takes the following form,

$$E(s) = \sum_{i \in S} g_i - X \tag{12}$$

Where X is a penalty that is included if capacitance constraints are violated. The above evaluation function is suitable for short-term tabu search. TS based on long-term memory requires historical information. In the next section we will show how a record of history of the moves is used to diversify search to improve results [16].

5.1 Diversification and Long Term Memory

In many combinatorial optimization problems, application of short-term memory alone may not produce a good solution. In [7], it is shown that the long-term memory functions can be very important for obtaining best results. In our work, we apply frequency-based diversification. In this strategy, we keep track of the number of times a certain move has been made. At the point when diversification is to be made, we penalize those moves that have been most frequent, thereby taking the search to areas unvisited thus far [7, 9, 10]. Therefore, the following modifications to short-term TS are done.

1. When the short-term TS algorithm hits a local optima, the following actions are taken:
   (a) All BiCMOS nodes are swapped to CMOS. Denote the number of those nodes as NUM_OF_BiCMOS
   (b) Search for least frequency nodes and replace them by BiCMOS. The search replacement process continues till the objective load threshold is reached or till the number of replaced nodes is equal to NUM_of_BiCMOS. Some of the BiCMOS nodes that were changed to CMOS may be changed again back to BiCMOS. Using the above, the search process is transferred to another region where the process might lead to a better solution.
   (c) Construct a new solution using the result of previous step.
2. Re-start the short-term memory component again and continue until another local optimum is reached; then repeat Step 1.

6 Results & Discussion

The approach described in this paper has been tested on several ISCAS-85 benchmark circuits. For the nine ISCAS-85 circuits used, the percentage of false paths reported ranged from 0-24%. Note that these are false paths from among the critical paths. In all but one case the maximum delay of the circuit did not change due to removal of false paths, but the number of gates on the sensitizable paths was reduced. For TS, two aspiration criteria (AS_1 and AS_2) were experimented with, and both produced similar results. All tools are developed in C and run on SUN SPARC workstations.

For all experiments, we used capacitance threshold to be 10% of total capacitance, the required reduction in the delay to be 30%. The tabu list size is an important parameter in TS. If the size is too small, the search will start cycling, and if it is too large, the search will be too restrictive. We experimented with several list sizes, ranging from 5 to 20. List sizes between 5 and 10 achieved best results in nearly all the benchmarks used.

Table 1 shows the results obtained by applying long-term memory component with customary aspiration criterion, AS_1. Tabu list size equal to 5 and candidate list size equal to 14 have been used. Diversification was applied after every 400 iterations. The number of BiCMOS gates required to speed up circuits using the presented approach is a very small percentage of the total number of gates. For medium to large sized circuits (greater than 200 gates) this number is less than 5%. For most circuits tested, a percentage delay reduction between 20% and 30% has been achieved with capacitance increase of less than 3%.

Table 2 shows a comparison of our work with that reported in [1]. In 3 out of four cases the result is a much better speed performance with a smaller number of BiCMOS gates. In all cases the percentage of BiCMOS gates used is lesser. Only for one circuit (c432), due to the constraint on total capacitance, the speed improvement was lesser, but so also was the number of BiCMOS gates used.

In this work, the candidate list of neighboring solutions is currently built using random moves. We are currently experimenting with evolutionary strategies where a node is selected for perturbation based on its fitness with respect to the overall objective of the search. This is expected to help the algorithm converge to better solutions in lesser time.

Acknowledgment

Authors acknowledge of King Fahd University of Petroleum and Minerals, Dhahran, for support.

References

[1] A. R. Baba-Ali and A. Bellaouar. An optimization tool for mixed CMOS/BiCMOS standard cells circuits. *Arabian Journal of Science and Engineering*, 19:4B:883–888, October 1994.

[2] J. Benkoski, E. Meersch, L. Claesen, and H. De Man. Timing verification using statically sensitizable paths. *IEEE Transaction on Computer-Aided Design*, 9(10):1073–1083, October 1990.

[3] D. Brand and V. Iyengar. Timing analysis using functional relationships. *Proceedings of ICCAD-86*, pages 126–129, 1986.

Circuit Name	Max Delay	Delay Reduction	% Of Delay Reduction	Total Capacitance	Capacitance Increase	% of Capacitance Increase	# of BiCMOS Gates
c432	171.911	43.342	25.20	109.260	1.478	1.00	19
c499	65.344	13.357	20.00	101.717	1.384	1.00	21
c880	125.506	25.089	20.00	215.111	2.148	1.00	23
c1355	109.860	26.615	24.20	399.564	7.242	2.00	63
c3540	185.201	48.645	26.30	683.401	1.796	0.26	18
struct	121.894	26.130	21.40	750.081	2.534	0.34	26
highway	32.438	8.815	27.20	15.15	0.762	5.00	16
fract	76.575	22.497	29.38	43.405	2.691	6.20	23

Table 1: Results of TS with long-term memory and aspiration criteria AS_1. (Delay in NS and capacitance in PF)

	Results of [1]		**Results of the proposed solution**	
Circuit Name	**% Speed Improvement**	**% of BiCMOS gates**	**% Speed Improvement**	**% of BiCMOS gates**
c432	40.9	11.1	25.2	4.8
c499	19.1	14.5	20.0	7.4
c880	12.3	21.7	20.1	2.7
c1355	11.2	9.4	24.3	4.3

Table 2: Comparison between results of [1] and our work.

[4] H. Chen and D. Du. Path sensitization in critical path problem. *IEEE Trans. on Computer-Aided Design of Integrated Cir. and Sys.*, 12(2):196–207, February 1993.

[5] H. Chen, D. Du, and L. Liu. Critical path selection for performance optimization. *IEEE Transactions on Computer-Aided Design*, 12(2):185–195, February 1993.

[6] D. Du, H. Yen, and S. Ghanta. On the general false path problem in timing analysis. *Proceedings of 26th Design Automation Conference*, pages 555–560, 1989.

[7] Fred Glover. Tabu search: A tutorial. *Technical Report, University of Colorado*, November 1990.

[8] S. Huang, T. Parng, and J. Shyu. A polynomial-time heuristic approach to approximate a solution to the false path problem. *30th ACM/IEEE Design Automation Conference*, pages 118–122, 1993.

[9] Roland Hubscher and Fred Glover. Applying tabu search with influential diversification to multiprocessor scheduling. *Computers Ops Res*, 21(8):877–884, 1994.

[10] Manuel Laguna and Fred Glover. Bandwidth packing: A tabu search approach. *Management Science*, 39(4):492–400, April 1993.

[11] S. Lin, M. Marek-Sadowska, and E. Kuh. Delay and area optimization in standard-cell design. *Proc. 27th Design Automation Conference*, pages 349–352, 1990.

[12] P. McGeer and R. Brayton. Efficient algorithm for computing the longest viable path in a combinational network. *26th ACM/IEEE Design Automation Conference*, pages 561–573, 1989.

[13] S. Perremans, L. Claesen, and H. De ManV. Static timing analysis of dynamically sensitizable paths. *26th ACM/IEEE Design Automation Conference*, pages 568–573, 1989.

[14] C. Reeves, editor. *Modern Heuristic Techniques for Combinatorial Problems.* Mc-Graw Hill Book Co., Europe, 1995.

[15] J. P. Roth. Diagnosis of automata failures: A calculus and a new method. *IBM J. Res. Develo.*, pages 278–281, October 1966.

[16] Sadiq M. Sait and H. Youssef. Iterative Computer Algorithms and Their Applications in Engineering. *IEEE CS Press, to appear*, 1998.

[17] S. Yen, D. Du, and S. Ghanta. Efficient algorithms for extracting the k most critical paths in timing analysis. *Proceedings of 26th Design Automation Conference*, pages 649–654, 1989.

[18] H. Youssef, Sadiq M. Sait, and Khalid Al-Farrah. Timing influenced force-directed floorplanning. *European Design Automation Conference with Euro-VHDL* **Euro-DAC'95**, *Brighton*, pages 156–161, September 1995.