# An Adaptive Interconnect-Length Driven Placer[†]

Chi-Ming Tsai, Kun-Tien Kuo, Chyi-Hui Hong, Rung-Bin Lin
*Department of Computer Engineering and Science*
*Yuan-Ze University, Chung-Li, Taiwan*
*Email: csrlin@cs.yzu.edu.tw*

## Abstract

*An adaptive interconnect-length driven standard cell placer (ILDP) is developed. The length bound for each source-sink pair is employed to direct the placement of each cell during recursive min-cut partitioning. Global migration, gate resizing, and buffer insertion are performed to make length bounds easier to satisfy. Bound re-computation is dynamically invoked to generate more realizable bounds based on the current partial placement. ILDP is integrated into a commercial tool set. Experimental results show more than 20% delay reduction can be achieved for some MCNC benchmark circuits.*

## 1. Introduction

Timing-driven placement has gone through a series of evolution, starting from net-weight-driven to net-length-driven, and to path-delay-driven. Net-weight-driven placement [1,2] assigns higher weights to critical nets to bias the positioning of a cell. The problem with this approach is that a non-critical path can easily become a critical one after layout. Path-delay-driven approach [3,4] checks the satisfaction of path delay requirements during a placement process. To handle path constraints easily, the delay of a path is formulated as a linear combination of timing variables of the nets and the logic gates on the path. This approach is very time consuming for a design with an enormous number of paths. Net-length-driven approach employs a set of net-length bounds to direct cell placement. It consists of two sub-problems, i.e., generating length bounds and positioning logic cells without violating length bounds. In [5], critical nets are given with net-length constraints prior to placement. A new min-cut algorithm driven by net-length bounds is proposed. In [6], a convex programming algorithm is used to compute the upper and lower bounds on net length. Macro-cells are placed at the appropriate locations to make the length of each net fall between its lower and upper bounds.

The work done in [7-9] emphasizes more on the generation of length bounds. The common procedure to generate net-length bounds for all the nets in a circuit works as follows. The slack of a path is first computed by taking only gate delays and timing requirements into account. Then, the path slack is distributed to the constituent nets of the path as net delay bounds. Path-slack distribution may take many iterations. Because net-delay bound can be converted into net-length bound based on a timing model, delay bound and length bound will be used interchangeably in this paper.

Other approaches to addressing interconnect delay problem include gate resizing, buffer insertion, and coupling-awareness physical design, etc. Gate resizing and buffer insertion can be performed pre-layout [10-12], during layout [13-16] or post layout [17-21]. Wire coupling can increase interconnect delay and produce a coupling noise. The increase in interconnect delay can elongate critical paths, whereas the coupling noise could possibly cause false switching on a device [22]. Because wire coupling is caused by the close proximity of wires that run in parallel for a long distance, researches in coupling-awareness physical design mainly focus on developing routing strategies to reduce the length of adjacent wires running in parallel [23-25].

A problem with net-length-driven approach is whether a placer can make all the nets (source-sink pairs) in a circuit satisfy their bounds. One approach to this problem is to compute interconnect bounds according to the circuit's electrical and physical characteristics [8]. The length bounds so obtained may be more realizable, but nets in a legal placement may sometimes still violate their bounds. Another approach is to re-compute bounds based on a bad layout and the placement task is performed again based on the new bounds. This approach is certainly very time consuming and may go forever. Yet another viable approach, like the one proposed in this paper, is to update bounds on the course of placement based on a partial placement. The delay bounds are adaptively modified according to the current status of the placement process. To go one step further, the proposed method employs global migration, gate

† This work was supported by National Science Council, Republic of China, under Grant No. NSC 89-2215-E-155-004

resizing, and buffer insertion to make delay bounds easier to satisfy. Experimental results show more than 20% delay reduction can be achieved for some MCNC benchmark circuits.

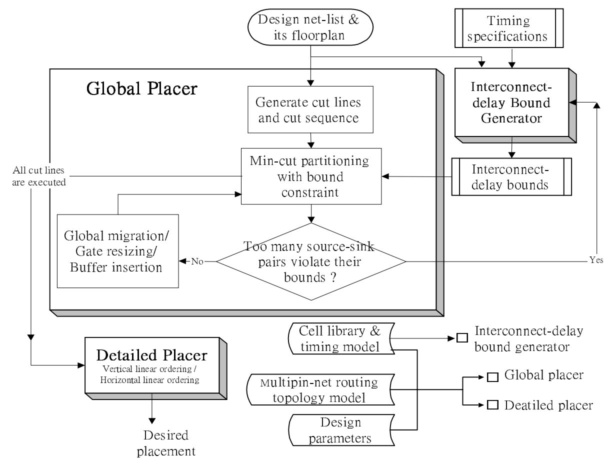
The rest of this paper is organized as follows. Section 2 details the methodologies used by *ILDP*. These include the approaches to delay bound computation, routing topology for a multi-pin net, min-cut partitioning, global migration, gate resizing, buffer insertion, bound re-computation, etc. Section 3 presents some experimental results. Section 4 draws a conclusion.

## 2. Interconnect-length driven placement

Figure 1 shows the proposed placement methodology, *ILDP*. The interconnect delay bound generator computes delay bounds for all the source-sink pairs in a circuit with considering the influence of functional false paths [26]. The global placer first generates the cut lines and determines the cut sequence, and then recursively performs circuit partitioning [27] until all the cut lines are executed. On the course of partitioning, the source-sink pair delay bounds are taken into account when one cell is moved from one partitioning region to another. If any of the source-sink pairs violates its bound, the cells connected by the underlying net are globally migrated from one partitioning region to another. This process is called global migration. Although global migration can reduce the distance between source-sink pairs, the delay bounds for some source-sink pairs might be still too tight to satisfy. In this situation, the global placer automatically performs gate resizing or buffer insertion to make the bounds easier to satisfy. Furthermore, delay bounds are dynamically re-computed based on the current partial placement when too many source-sink pairs violate their bounds. As the partitioning process goes, the partial placement will give more accurate cell positions such that the new delay bounds will be more realizable. Finally, detailed placement decides the exact cell positions to minimize the total wire length.

### 2.1. Net-length bound generator

The bound generator, though employing *MIMP* [8] as a basis, can distribute path slack to the constituent source-sink pairs of a path instead of the constituent nets. It can also assign a minimal bound to each source-sink pair based on the cells connected to the underlying net. In general, we would like a critical source-sink pair as shorter as possible. However, from the bound realization viewpoint, we would like a bound as larger as possible. In this work, the weights used in distributing path slacks can support both viewpoints and are chosen to be a function of fanout. Other parameters such as driving capability can be used along with fanout to derive the



Design net-list & its floorplan
Timing specifications
Global Placer
Generate cut lines and cut sequence
Interconnect-delay Bound Generator
All cut lines are executed
Min-cut partitioning with bound constraint
Interconnect-delay bounds
Global migration/ Gate resizing/ Buffer insertion
No
Too many source-sink pairs violate their bounds ?
Yes
Cell library & timing model
Interconnect-delay bound generator
Detailed Placer
Vertical linear ordering / Horizontal linear ordering
Multipin-net routing topology model
Global placer
Deatiled placer
Design parameters
Desired placement

**Figure 1.** Overview of ILDP

weights. The bound generator calculates gate delay using a non-linear delay model [28] in terms of input slope and output load. To improve the efficiency of bound computation, the *PCL* [29] (path constraint list) created during false path identification is employed to exclude false paths during bound computation. Basically, a delay bound generator should take the coupling delay into account, but it is not necessary for us to do so in placement.

For initial bound computation prior to placement, the capacitance derived from the minimum delay bound is used to adjust gate delays to make delay bounds more accurate. When bound re-computation is performed during min-cut partitioning, gate delays should be recalculated due to the change of wire capacitance derived from a partial placement. An interconnect delay is computed for each source-sink pair by using the distributed RC Elmore delay model. Timing analysis takes the estimated interconnect delay into account when it calculates the latest arrival time for each source-sink pair. If the slack of the longest path traversing the source-sink pair is positive, *MIMP* algorithm is employed to distribute the remaining slack. The final delay bound is taken as the sum of the estimated interconnect delay and the newly assigned slack. Otherwise, negative slack should be distributed to each source-sink pair $f$ on the underlying path according to the following rule:

*If the estimated delay of $f$ is smaller than the old bound,*

$$bound_{new}(f) = \text{estimated delay of } f \text{ ; otherwise,}$$

$$bound_{new}(f) = \text{estimated delay} + path\_slack \times violation_{f} \, / \, violation_{total} \, .$$

Where $violation_f$ is the amount of bound violation on $f$ and $violation_{total}$ is the total amount of bound violations on the longest path traversing $f$. Redistributing negative path slack in this way can have the following effects:

(a). A source-sink pair that does not violate its bound will continue to satisfy the new bound during subsequent partitioning.

(b). A source-sink pair that has a small amount of violation will be very likely to satisfy the new bound during subsequent partitioning.

(c). A source-sink pair that has a large amount of violation will very likely continue to violate the new bound so that global migration, gate resizing or buffer insertion is invoked to eliminate the violation during subsequent partitioning.

Our bound generator is efficient, for example, the benchmark circuit s38584 with 11392 cells taking only 185 seconds on a SUN UltraSparc 10 with 128 Mb memory.

## 2.2. Multi-pin net topology and timing model

A multi-pin net routing problem can be solved by constructing an optimal *rectilinear Steiner tree* (RST). Because it is an NP-hard problem [30], the problem is approximated by using a simpler routing topology model shown in Figure 2. Given the terminal points of a net, the minimum bounding box enclosing all the terminal points is first created. Suppose the longer side of the bounding box is in the horizontal direction. Then, a routing tree is formed by drawing a trunk (i.e., $\overline{AB}$) across the center of the shorter side and by drawing vertical wire segments to connect all the terminal points to the trunk. Clearly, the maximum length of a $k$-pin net is not greater than $L + 0.5kW$ and the estimation for a two-pin net is optimal. Here, we assume that a pin is located at the center of a partitioning region.

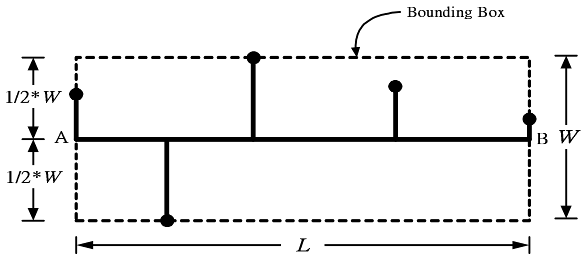


**Figure 2.** A multi-pin net routing topology

To compute the RC delay for each source-sink pair, the topology model is employed to create a routing tree whose root and leaves represent respectively the source node and the sink nodes of a net. Each wire segment is then modeled as a $\pi$-type circuit and the interconnect delay is estimated based on the distributed RC Elmore delay model.

## 2.3. Global placer

The global placer places logic cells at the various regions such that the total wire length is minimized and each source-sink pair can satisfy its delay bound. Figure 3 shows the global placement algorithm. V*iolation($n_i$)* denotes the bound violation of net $n_i$, which is the sum of the bound violations of the source-sink pairs that constitute net $n_i$, and $\varepsilon$ is a user-specified threshold. The global placer performs the following main tasks: min-cut partitioning, global migration, gate resizing, buffer insertion, and invoking delay bound re-computation.

```
Generate cut lines and determine a cut sequence;
For each cut line {
  Perform min-cut partitioning with bound constraints;
  If ( too many source-sink pairs violate their delay bounds )
     Perform bound re-computation;
  Else {
    For each net $n_i$, if any of its source-sink pairs violates the bound {
    Perform global migration;
    if ( 0< $violation(n_i)$ <$\varepsilon$ )
    Apply gate resizing;
    Else
    Perform buffer insertion;
    }}
}
```

**Figure 3.** Global placement

**2.3.1. Min-cut partitioning.** Min-cut partitioning, in addition to satisfying size constraints and maximizing gains, has to consider length bounds when moving a cell to another partitioning region. Whether a selected cell *m* can be moved is determined by the following criteria:

(a). If $violation\_diff(m) \leq 0$, then cell *m* can be moved to another region.

(b). If $violation\_diff(m) > 0$ and $gain(m) \leq 0$, then cell *m* must be kept in the original region.

(c). If $violation\_diff(m)>0$, $gain(m)>0$, and $\omega \times gain(m) - (1-\omega) \times violation\_diff(m) > 0$, then cell *m* can be moved to another region; otherwise, cell *m* is kept in the original region.

Where *gain(m)* and *violation_diff(m)* are respectively the cut-size reduction and the change of total bound violations for moving cell *m* to another region. A net has a violation if any of the source-sink pairs of this net violates its delay bound. The bound violation of this net is computed as the sum of the bound violations of the source-sink pairs that constitute this net. By varying $\omega$, $0 \leq \omega \leq 1$, we can adjust the role of delay bounds during min-cut partitioning. Note that criterion (c) allows a source-sink pair to violate its delay bound when the gain is considerably large. The violation will be eliminated later by global migration, gate resizing, or buffer insertion.

**2.3.2. Global migration.** The global placer could migrate the cells connected by a net to other partitioning regions in order to eliminate a violation. Moving these cells to another partitioning regions may make some other source-sink pairs connected to these cells also violate their delay bounds. So, *violation_diff(m)* defined above is used to determine whether a cell *m* can be moved. If it is smaller than zero, cell *m* can be moved to the desired region. Global migration is terminated when the violation is eliminated or all of the underlying cells can not be moved.

**2.3.3. Gate resizing.** When some source-sink pairs can't satisfy their delay bounds, gate resizing is tried to solve this problem. Driving gate and/or the driven gates can be resized to increase delay bounds. We consider only driving gate resizing, which is done by using a gate with higher driving capability. Although scaling up a driving gate can increase the delay bounds of its source-sink pairs, the delay bounds of the source-sink pairs driven by the preceding gate would be decreased because the gate capacitance of the driving gate has already increased. Theoretically, gate-resizing should be conducted backward toward the source of a path. But if the increase in the driving gate capacitance is much smaller than the wiring capacitance of a net, we do not have to perform backward gate resizing. In fact, if we have to do so, it probably indicates that buffer insertion is required.

**2.3.4. Buffer insertion.** When a buffer is inserted into a long net, three important issues must be addressed: where to insert the buffer, how large the buffer size is and how to re-distribute the original delay bound to the newly created source-sink pairs. The insertion position is selected as the point along the most critical source-sink path on a routing tree such that bound violations can be greatly reduced. The size of the inserted buffer depends on its output load capacitance. In general, the greater its output load capacitance, the larger is its size. But a buffer of proper size should be used just enough to eliminate bound violations. After buffer insertion, a net is separated into two shorter nets and bound redistribution should be made based on the buffer's position and driving capability.

**2.3.5. Delay bound re-computation.** When the number of bound violations is larger than a user-specified threshold, it indicates that the global placer with the current set of delay bounds will not be able to obtain a timing-satisfied design. In this situation, the global placer stops the placement process and bound re-computation is invoked. The information passed from the global placer to the bound generator includes the estimated delay of each source-sink pair derived from the current partial placement. The bound generator uses this information to generate a set of more realizable bounds. Then, the global placer continues the placement process with these new bounds. If it goes well, the frequency of bound re-computation should be limited.

### 2.4. Detailed placer

Detailed placement decides the exact positions for the cells located in each of the partitioning regions. Since each cell is located within a small region, length bounds are no longer used here to direct the positioning of a cell. The detailed placement performs a two-phase linear ordering [31]. The first phase divides all the cells within a partitioning region into *r* subsets, where *r* is the number of cell rows within this region. This phase minimizes the number of wires among rows in a region, whereas the second phase minimizes the total wire length on the same row.

## 3. Experimental results

*ILDP* is integrated into a commercial tool set called *CMT.* The placements obtained respectively from *ILDP* and *CMT placer* are routed by *CMT router*. Post-layout resistance and capacitance are used to compute the interconnect RC delay. The interconnect delay and cell delay are employed to produce a SDF (Standard Delay Format) file which is used by *Synopsys Design Time* to perform post-layout timing analysis.

In order to make interconnect delay more significant, we increase the sheet resistance of each routing layer such that a small increase in interconnect length would significantly increase the interconnect delay, but would slightly increase the gate delay. The purpose is to see how effective *ILDP* can manage interconnect length. Three *MCNC* benchmark circuits, s5358, s13207, and s38584 are employed for this study. These circuits are synthesized by Synopsys Design Compiler with an in-house 0.25um cell library. The results are presented in Tables 1, 2, and 3, where placement strategy indicates what kind of weighting method is used in bound computation, whether global migration is applied, or whether gate resizing is employed. *NI* indicates that the weight of the source-sink pairs that constitute a net is set proportional to the square root of the fanout of this net, whereas *Inv* indicates that the weight is set proportional to the inverse of the square root of the fanout. A path between memory cells is called a memory path, whereas a path starting from or ending at an IO pin is called an IO path. "√" indicates the associated placement strategy is employed, whereas "×" indicates otherwise. The longest path delay produced by *CMT* is given in the last row.

In Tables 1 and 2, the delays of the longest memory paths and the longest IO paths produced by *ILDP* are smaller than that produced by *CMT.* It is found for s5378 that the placement results produced by *ILDP* employing *NI* strategy are better than that by *ILDP* employing *Inv* strategy. But, the phenomenon is opposite for s13207. This reminds us of the argument that a critical source-sink pair should be routed as shorter as possible, but a bound should be made as larger as possible from the bound

**Table 1.** The longest path delays of s5378 after routing

| *ILDP* Placement Strategy | | | Longest Memory Path | | Longest IO Path | |
|---|---|---|---|---|---|---|
| Weighting method | Global Migration | Gate Resizing | Delay (ns) | Delay improved | Delay (ns) | Delay improved |
| *NI* | × | × | 5.36 | 29.4 % | 5.18 | 0.2 % |
| *Inv* | × | × | 5.40 | 28.9 % | 4.92 | 5.2 % |
| *NI* | √ | × | 4.91 | 35.3 % | 4.30 | 17.2 % |
| *Inv* | √ | × | 5.14 | 32.3 % | 4.86 | 6.4 % |
| *NI* | √ | √ | 4.88 | 35.7 % | 3.84 | 26.0 % |
| *Inv* | √ | √ | 5.54 | 27.0 % | 4.58 | 11.8 % |
| *CMT* | | | 7.59 | 0 % | 5.19 | 0 % |

**Table 2.** The longest path delays of s13207 after routing

| *ILDP* Placement Strategy | | | Longest Memory Path | | Longest I/O Path | |
|---|---|---|---|---|---|---|
| Weighting method | Global Migration | Gate Resizing | Delay (ns) | Delay improved | Delay (ns) | Delay improved |
| *NI* | × | × | 12.05 | 9.9 % | 7.36 | 24.4 % |
| *Inv* | × | × | 11.41 | 14.7 % | 6.43 | 34.0 % |
| *NI* | √ | × | 10.81 | 19.2 % | 6.35 | 34.8 % |
| *Inv* | √ | × | 10.65 | 20.4 % | 5.82 | 40.3 % |
| *NI* | √ | √ | 9.96 | 25.6 % | 7.18 | 26.3 % |
| *Inv* | √ | √ | 9.31 | 30.4 % | 5.76 | 40.9 % |
| *CMT* | | | 13.38 | 0 % % | 9.74 | 0 % |

**Table 3.** The longest path delays of s38584 after routing

| *ILDP* Placement Strategy | | | Longest Memory Path | | Longest I/O Path | |
|---|---|---|---|---|---|---|
| Weighting method | Global Migration | Gate Resizing | Delay (ns) | Delay improved | Delay (ns) | Delay improved |
| *NI* | × | × | 32.02 | -59.2 % | 47.86 | 23.6 % |
| *Inv* | × | × | 32.31 | -60.6 % | 48.08 | 23.3 % |
| *NI* | √ | × | 33.35 | -65.8 % | 38.23 | 39.0 % |
| *Inv* | √ | × | 27.89 | -38.6 % | 52.18 | 16.8 % |
| *NI* | √ | √ | 24.26 | -20.6 % | 49.18 | 21.5 % |
| *Inv* | √ | √ | 29.89 | -48.6 % | 47.57 | 24.1 % |
| *CMT* | | | 20.12 | 0 % | 62.68 | 0 % |

realization viewpoint. Because these two viewpoints contradict to each other, it is reasonable to expect that *NI* strategy can not totally outperform *Inv* strategy or vice versa.

In Table 3, the longest IO path delays produced by *ILDP* are smaller than the one produced by *CMT.* But the longest memory path delays produced by *ILDP* are longer than that produced by *CMT.* This is because the IO paths of s38584 are much longer than the memory paths. As a consequence, the delay bounds of the source-sink pairs on the IO paths are tighter than that on the memory paths. Therefore, *ILDP* would tend to minimize the length of each source-sink pair on the IO paths while make the delays of the memory paths satisfy the timing specifications.

Ideally, the longest path delay produced by *ILDP* employing both global migration and gate resizing should be smaller than that by *ILDP* simply employing global migration; the longest path delay produced by *ILDP* employing only global migration should be smaller than that by *ILDP* employing neither global migration nor gate resizing. For s5378 and s13207, this is almost true except for a few special cases, but it is not true for s38584. The possible reason for this is that we can not control the routing process so that the actual interconnect delay after routing may be very far from the one predicted by *ILDP*.

The results obtained by *ILDP* with buffer insertion in Table 4 show that buffer insertion can further reduce the longest path delay. Here, *NI* strategy is used to derive weights for computing delay bounds. The number of buffers inserted is quite limited, at most 5% of the total cell area. Note that *CMT* does not perform any buffer insertion. For a fair comparison, we are currently working weighing of the gain and bound violation during min-cut partitioning is not properly performed. Overly emphasizing bound violation can easily cause a routability problem. We guess that this problem can be improved by using a more robust multi-way partitioning algorithm [32].

**Table 4.** The longest path delay and total cell area

| Circuits | Number of cells | Number of nets | Number of IO pins | *CMT* | | *ILDP* | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Longest path delay | Total cell area (um^2) | Longest path delay | Delay improved | Number of Inserted buffers | Total cell area (um^2) | Area increased |
| S13207 | 3006 | 3367 | 155 | 13.32 ns | 201130 | 8.12 ns | -39.0% | 18 | 202060 | +0.46% |
| S15850 | 3767 | 4124 | 104 | 14.55 ns | 221940 | 10.76 ns | -26.1% | 32 | 223630 | +0.76% |
| S38417 | 11110 | 11750 | 137 | 21.03 ns | 666340 | 15.49 ns | -26.3% | 359 | 684810 | +2.77% |
| s35932 | 11544 | 12900 | 358 | 59.15 ns | 693190 | 40.30 ns | -31.9% | 339 | 710990 | +2.57% |
| s38584 | 11392 | 12536 | 293 | 54.74 ns | 642170 | 35.37 ns | -35.4% | 616 | 674180 | +4.98% |

## 4. Conclusion

An Interconnect-Length Driven standard cell Placer (*ILDP*) employing source-sink pair delay bounds to direct the positioning of cells during recursive min-cut partitioning has been developed. Global migration, gate resizing, and buffer insertion are performed to make delay bounds easier to satisfy. When too many source-sink pairs violate their bounds during min-cut partitioning, more realizable delay bounds are dynamically derived from the current partial placement. *ILDP* is integrated into a commercial tool set. Results from *MCNC* benchmark circuits show more than 20% delay reduction can be achieved. In the future we will apply delay bounds to wire sizing during placement.

## 5. References

[1] A. E. Dunlop, V. D. Agrawal, D. N. Deutsch, M. F. Jukl, P. Kozak, and M. Wiesel, "Chip layout optimization using critical path weighting," the 21th ACM/IEEE DAC, 1984, pp. 133-136.

[2] M. Burstein and M. N. Youssef, "Timing influenced layout design," the 22th ACM/IEEE DAC, 1985, pp. 124-130.

[3] T. Koide, M. Ono, S. Wakabayashi, Y. Nishimaru, and N. Yoshida, "A new performance driven placement method with the Elmore delay model for Row Based VLSIs," the 32th DAC, 1995, pp. 405-412.

[4] W. Donath et al., "Timing driven placement using complete path delay," the 27th ACM/IEEE DAC, 1990, pp. 84-89.

[5] M. Terai, K. Takahashi, and Koji Sato, "A new min-cut placement algorithm for timing assurance layout design meeting net length constraint," the 27th ACM/IEEE Design Automation Conference, 1990, pp. 96-102.

[6] T. Gao, P. M. Vaidya, and C. L. Liu, "A performance driven marco-cell placement algorithm," the 29th ACM/IEEE DAC, 1992, pp. 147-152.

[7] R. Nair, C. L. Berman, P. S. Hauge, and E. J. Yoffa, "Generation of performance constraints for layout," IEEE Trans. on CAD, Vol. 8, Aug. 1989, pp. 860-874.

[8] H. Youssef, R.-B. Lin, and E. Shragowitz, "Bounds on net delays for VLSI circuits," IEEE Trans. on Circuits and Systems – II: Analog and Digital Signal Processing, Vol. 39, No. 11, Nov. 1992, pp. 815-824.

[9] Wing K. Luk, "A fast physical constraint generator for timing driven layout," the 28th ACM/IEEE DAC, 1991, pp. 626-631.

[10] K. S. Lowe, and P. G. Gulak, "A joint gate sizing and buffer insertion method for optimizing delay and power in CMOS and BiCOMS combinational logic," IEEE Trans. on CAD, Vol. 17, May 1998, pp. 419-434.

[11] H.-R. Lin and T.-T. Hwang, "Power reduction by gate sizing with path-oriented slack calculation," the 32th ACM/IEEE DAC, 1995, pp. 7-12.

[12] P. Girard et al., "A gate resizing technique for high reducition in power consumption," Proc. of Low Power Electronics and Design, Aug. 1997, pp. 281-286.

[13] Y. Jiang, S. S. Sapatnekar, C. Bamji and J. Kim, "Interleaving buffer insertion and transistor sizing into a single optimization," IEEE Transactions on VLSI Systems, Vol. 64, Dec.1998, pp. 625-633.

[14] Y. Jiang, S. S. Sapatnekar, C. Bamji and J. Kim, "Combined transistor sizing with buffer insertion for timing optimization," CICC, pp. 605-608.

[15] Y. P. Chen and D. F. Wong, "An algorithm for zero-skew clock tree routing with buffer insertion," Proc. of European Design and Test Conf., 1996, pp. 230-236.

[16] J. Lillis, C.-K. Cheng, and T-T Y. Lin, "Simultaneous routing and buffer insertion for high performance interconnect," the Sixth Great Lakes Symposium on VLSI, 1996, pp. 148-153.

[17] K.-J. Huang and Y.-L. Lin, "Post-layout performance optimization using gate sizing and buffer insertion," 8th VLSI Design/CAD Symposium, Sun-Moon Lake, Taiwan, Aug. 1997, pp. 257-268.

[18] J. Lillis, C.-K. Cheng and T.-T. Y. Lin, "Optimal and efficient buffer insertion and wire sizing," CICC, 1995, pp. 259-262.

[19] J. Kim, Y.-C. Hsu and D. H. C. Du, "A new gate selection method for resizing to circuit performance optimization," IEEE International Symposium on Circuits and Systems, Vol. 4, 1996, pp. 461-464.

[20] J. Kim and D. H. C. Du, "Performance optimization by gate sizing and path sensitization," IEEE Trans. on CAD, Vol. 17, May 1998, pp. 459-462.

[21] C.-C. Tsai, D.-Y. Kao and C.-K. Cheng, "Performance driven bus buffer insertion," IEEE Trans. on Computer-Aided Design, Vol. 15, April 1996, pp. 429-436.

[22] L. Gal, "On-chip cross talk-the new signal integrity challenge," CICC, 1995, pp. 251-254.

[23] M. Sengupta, S. Lipa, P. Franzon, and M. Steer, "Crosstalk driven routing advice," The 44th Electronic Components and Technology Conference, May 1994, pp. 687-694.

[24] H. P. Tseng, L. Scheffer, and C. Sechen, "Timing and crosstalk driven area routing," The ACM/IEEE Design Automation Conference, June 1998, pp. 378-381.

[25] T. Xue, E. S. Kuh, and D. Wang, "Post global routing crosstalk synthesis," IEEE Trans. on CAD, Vol. 16, No. 12, Dec. 1997, pp. 1418-1430.

[26] H. C. Chen and H. C. Du, "Path sensitization in critical path problem," IEEE Trans. on CAD, vol. 12, Feb. 1993, pp. 196-207.

[27] C. M. Fiduccia and R. M. Mattheyses, "A linear time heuristic for improving network partitions," the 19th ACM/IEEE DAC, 1982, pp. 175-181.

[28] "Chapter 8. CMOS Non-linear Timing Model," *Synopsys, Library Compiler Ref. Manual*, v1.

[29] C. Oh and M. Ray Mercer, "Efficient logic-level timing using constraint-guided critical path search," IEEE Trans. on VLSI Systems, vol. 4, Sept. 1996, pp. 346-355.

[30] M. R. Garey and D. S. Johnson, "The rectilinear Steiner tree problem is NP-complete," SIAM J. Appl. Math., 1996, pp. 255-265.

[31] E. Shragowitz, L. Lin and S. Sahni, "Models and algorithms for structured layout," CAD, Vol. 20, No. 5, June 1988, pp. 263-271.

[32] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel hypergraph partitioning: Applications in VLSI domain," IEEE Trans. on VLSI Systems, Vol. 7, No. 1, March 1999, pp. 69-79.