Simultaneous Wire Sizing and Wire Spacing in Post-Layout Performance Optimization

Jiang-An He

Silicon Valley Research, Inc.,
San Jose, CA 95112

Tel: (408) 361-1076
Fax: (408) 361-0330
e-mail: he@svri.com

Hideaki Kobayashi

Dept. of Electrical and Computer Engineering,
University of South Carolina
Columbia, SC 29208
Tel: (803) 777-6953
Fax: (803) 777-8045
e-mail: kobayash@ece.scarolina.edu

Abstract - In this paper, we study the wire sizing and wire spacing problem for post-layout performance optimization under Elmore delay model. Both ground capacitance and coupled capacitance in a wire are included in interconnect delay calculation. Combined with general ASIC design flow, we construct section constraint graph in each routing region and use the graph to guide segment sizing and spacing. By defining a cost function to trade-off between interconnect delay and routing area, we formulate wire sizing and wire spacing problem (WSSP) into a constraint-optimization problem and develop a heuristic algorithm to solve it. The preliminary experiments are promising.

1. Introduction

As the result of scaling to submicron and deep-submicron dimensions, the timing issue is becoming more and more important in determining circuit performance and reliability. The trend in VLSI process with smaller wire width, smaller space, and larger wire aspect ratio (thickness/width) results in more significantly coupled capacitance to adjacent neighbors. This also make the interconnect delays to increasingly dominate the performance[12].

The interconnect delay is caused by capacitance and resistance of wire segments (we do not consider inductance in this paper). The capacitance for the wire can be classified into two groups: ground and coupled. Since in the future technology the coupled capacitance for a pair of unit length metal wires will be much larger than the ground capacitance, the coupled capacitance will become very important in the interconnect delay[7]. Current research related to layout synthesis does not address this problem sufficiently.

There are several techniques in improving circuit performance, which include buffer insertion, routing tree topology optimization, and wire sizing[1-6]. Among these techniques, wire sizing (selectively widening some of the wires in a routing tree) technique has gotten a lot of attentions in recent years. In [1], some important properties for wire sizing were developed to construct algorithms for minimizing delays. The optimal wire sizing formula for a wire under Elmore delay model was also developed in [3]. In [7], the wire sizing problem for multiple sources was studied. In reviewing all the existing wire sizing studies, some drawbacks exist, which include: a) coupled capacitance was not explicitly considered; b) minimizing delays instead of meeting timing constraints was the goal; c) routing congestion was not considered; and d) design methodologies were not discussed.

On the other hand, some studies on wire spacing technique (adjusting wire space between wires) have been conducted to reduce coupled capacitance for solving crosstalk and performance issues[8-10]. However, the existing studies only considered spacing technique independently.

In practice, both wire sizing and wire spacing techniques can improve circuit performance and reliability. Meanwhile, they also need extra area resource, which is another important factor in optimizing circuit design. Therefore, trade-off between performance and routing area is a key for design success. This paper uses both wire sizing and wire spacing techniques to optimize routing area and interconnect delay in post-layout phase. Elmore delay model[11] is used in interconnect delay calculation. We propose a design step between detailed routing and compaction to do wire sizing and wire spacing for all critical nets with timing constraints. The proposed algorithms can be used for normal ASIC design flow. Instead of handling wire sizing and wire spacing independently, we combine them together. A very important idea in optimizing interconnect delay in this paper is that we do not minimize the interconnect delay, but we expect that the interconnect delay is not too far away from its given bound. This strategy provides a practical trade-off in dealing with timing and area. By constructing a cost function, we formulate the wire sizing and wire spacing problem as a constraint-optimization problem and solve it with a heuristic algorithm.

The rest of this paper is organized as follows. In Section 2, we introduce the design flow where the wire sizing and wire spacing techniques will be used, give definitions for some basic terminologies, and formulate wire sizing and wire spacing problems. In Section 3, the delay models and the calculation of interconnect delays considering both ground capacitance and coupled capacitance are described. Also, delay sensitivities to wire sizing and wire spacing are analyzed in the same section. In Section 4, we introduce section constraint graph, describe wire sizing and wire spacing techniques under this graph, and develop a heuristic algorithm to solve the wire sizing and wire spacing problem. The algorithm analysis is also included in the same section. Section 5 presents some preliminary experiments followed by conclusions and future work in Section 6.

2. Problem Definitions

2. 1 Design flow

In this paper, we consider cell-based layout synthesis with channel-based routing. As shown in Fig. 1(a), the conventional design flow consists of placement, global routing, detailed routing (channel routing most likely), and compaction. Since wire sizing and wire spacing techniques need accurate wiring information and the design flow must provide flexibility to take wire sizing and wire spacing results for further adjustment, the reasonable place to do it is between detailed routing and compaction. Based on this flow, we propose an extra-step, VGEOMETRY, between the detailed routing (all channels are routed first) and compaction, to perform wire sizing and wire spacing features in order to optimize chip area and meet timing constraints. Fig. 1(b) shows the proposed design flow.

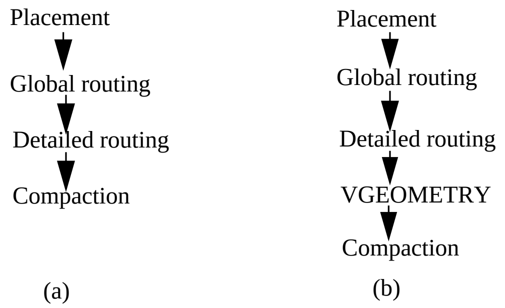

Fig. 1: (a) Conventional design flow.
(b) Proposed design flow with VGEOMETRY.

2.2 Basic concepts in wire sizing and wire spacing

When two parallel segments have overlapped length and there is no other segment between their overlapped area, we say the two segments are ***adjacent***. We use ***adj(e)*** to describe the adjacent segments for segment ***e***. For example, segment e0 in Fig.2 has four adjacent segments, which is: adj(e0) = {e1, e2, e3, e4}.

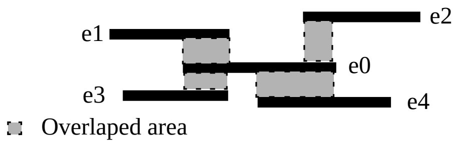

Fig. 2. Adjacent segment.

Each segment has a width (we only consider uniform sizing for a segment in this paper), which is also called ***size*** a lot of times. The side distance between a segment e and each of adj(e) is defined as ***space***. So, for a segment, its space may not be a single value but a set of values depending on its adjacent segments. Wire sizing technique is to assign size to each segment and wire spacing technique is to assign space to each pair of adjacent segments.

In practice, wire size and wire space have some limits decided by certain technologies. It is reasonable to assume that the effective sizing and spacing ranges are bounded by W = { w0, w1, ..., wr) and P = {p0, p1, ..., pm), respectively. To incrementally increase size and space, we define a minimum step, Δ ,for wire sizing and spacing. So, the bounded ranges can be represented as W = {w0, w0+ Δ , ..., w0+r* Δ) and P = {p0, p0+ Δ , ..., p0+m* Δ).

2.3 Partitioning of routing regions

For a given routing region such as a channel, we can partition it into a set of sub-regions and we call these sub-regions as ***sections***. Fig. 3 is an example of a partitioned channel. The reason we partition channels into sections is to control the resolutions of segments. Because each segment has uniform size, the partitioning can have more control on segment lengths, which in turn leads to non-uniform sizing for a net.

Cut lines are used to cut a routing region into sections. Depending on the distance between cut lines, we can control the resolution of wire sizing and wire spacing. In a routing tree, we define each horizontal or vertical segment as a ***routing segment*** and the intersection between horizontal segment and vertical segment as a ***routing node***. Also, we define the intersection between cut line and routing segment as a ***section node***. With the introduction of section node, we define a segment between any two nodes (routing nodes or section nodes) as a ***operating segment***.

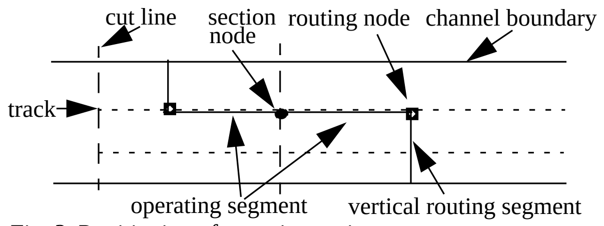

Fig. 3. Partitioning of a routing region.

2.4 *Problem formulation*

Given a detailed routing and partitioning of routing regions, we can extract all operating segments for each net and their adjacent segments. Our interest is for the horizontal operating segments, which will be used in wire sizing and wire spacing. The major reasons we do not consider vertical operating segments are: a) vertical operating segments are usually much shorter in a channel; b) sizing and spacing vertical operating segments may easily cause DRC problem.

The **wire sizing and spacing problem (WSSP)** is defined as: given a detailed routing, partitioning of routing regions, and timing constraints for critical nets, find a solution with assigned size and space for horizontal operating segments to meet timing and area targets.

For a net n, its routing tree is given by the detailed routing. Let n has pin set (s0, s1, ..., sk), where s0 is the source and other pins are sinks (we do not handle multiple sources here). For each sink si, we assume its ***required arrival time*** is given as ***q(i)***. If we define ***path(i)*** as the path from s0 to si and ***d(i)*** as the actual delay to si along the path, then we have the ***delay slack*** on si as ***q(i) - d(i)***, denoted by ***ds(i)***. If ds(i) is positive, we say the path meets its timing constraints and it is a ***legal path***; otherwise the path violates its timing constraints and it is a ***illegal path***. For a critical net, if it does not have any illegal path, then it meets timing constraints and it is a ***legal net***; otherwise it is an ***illegal net***.

We define a ***distance function*** for a net to measure how far it is away from legal status. For net ***n***, we use ***dist(n)*** to represent its distance and ***ip(n)*** to represent the set of illegal paths in n. If n is a legal net, then dist(n) = 0; otherwise, dist(n) is defined as:

$$dist(n) = \left(\sum_{i \in ip(n)} ds(i)\right) / |ip(n)|$$

Obvious, for illegal net n, dist(n) < 0. The smaller dist(n), the farther the net is from legal status.

A cost function to balance area and timing requirements for a given problem ***WP*** and a solution ***S*** is defined as:

$$\cos t(WP, S) = \left(\sum_{i \in CR(WP)} dist(i)\right) \bullet \lambda 1 - h \bullet \lambda 2$$

where ***CR(WP)*** is the set of critical nets in ***WP*** and ***h*** is the current total height of channels. Two constants $\lambda 1$ and $\lambda 2$ are used to trade-off timing and area (we set $\lambda 1 + \lambda 2 = 1$ and $\lambda 1 \geq 0,\ \lambda 2 \geq 0$).

Based on above definitions, we can formulate WSSP as a constraint-optimization problem:

Objective:
maximize (cost(WP, S))
Constraints:
size constraints (w0, w0+ Δ , ..., w0+r* Δ)
space constraints (p0, p0+ Δ , ..., p0+m* Δ)

3. Timing analysis

A routing topology of a net can be formulated as a distributed RC network. We use Elmore delay model and standard RC model in delay calculation for interconnects and drivers, respectively.

3.1 Delay calculation

For a segment ***e***, its Elmore delay can be calculated as:

$$d(e) = r(e) \bullet \left(\frac{c(e)}{2} + C(Te)\right)$$

where ***d(e)***, ***r(e)***, and ***c(e)*** represent delay, resistance, and capacitance on e, respectively. ***Te*** is the subtree under ***e*** and rooted at ***e***-entered node. ***C(Te)*** is the lumped capacitance in subtree Te.

The delay of a driver ***g*** can be calculated as:

$$d(g) = di(g) + Rd(g) \bullet CL(g)$$

where ***d(g)***, ***di(g)***, ***Rd(g)***, and ***CL(g)*** represent delay, intrinsic delay, driver resistance, and load on ***g***, respectively.

For a pair of adjacent segments ***i*** and ***j***, the coupled capacitance between them can be calculated as [8, 13]:

$$cx(i, j) = Kc \bullet l(i, j) \bullet \left(\frac{1}{p(i, j)^{1.34}}\right)$$

where ***cx(i,j)***, ***l(i,j)***, and ***p(i,j)*** represent the coupled capacitance, overlapped length, and space between i and j, respectively. Kc is a technology-dependent constant.

Let r0, c0 represent sheet resistance and unit area capacitance. For segment e, we have:

$$r(e) = \alpha \bullet r0 \bullet (l(e))/(w(e))$$
$$cg(e) = \beta \bullet c0 \bullet l(e) \bullet w(e)$$
$$c(e) = cg(e) + cx(e)$$

where ***cg(e)*** is the ground capacitance and ***cx(e)*** is the coupled capacitance on segment e. α and β are two technology-dependent constants.

Now assume segment ***e*** has the same space ***p*** with all its adjacent segments. and the total overlapped length is ***lx(e)***. Then the Elmore delay on segment ***e*** can be transformed to:

$$d(e) = T1 + \frac{T2}{w(e)} + \frac{T3}{w(e) \bullet p^{1.34}}$$

where ***T1***, ***T2***, and ***T3*** are three constants for ***e*** and can be calculated by:

$$T_1 = \frac{\alpha \bullet \beta \bullet r_0 \bullet c_0 \bullet l(e)^2}{2}$$
$$T_2 = \alpha \bullet r_0 \bullet l(e) \bullet C(T_e)$$
$$T_3 = \frac{\alpha \bullet r_0 \bullet K_c \bullet l(e) \bullet lx(e)}{2}$$

Based on this calculation, we can get ***delay sensitivity*** of segment ***e*** on wire size ***w(e)*** and wire space ***p*** as:

$$\Delta d(e) = -\left(\frac{T_2 + \frac{T_3}{p^{1.34}}}{w(e)^2} \bullet \Delta w(e) + \frac{1.34 \bullet T_3}{w(e) \bullet p^{2.34}} \bullet \Delta p\right)$$

We can see that increasing wire size *w(e)* or space *p* can reduce delay on the segment. But the reduced amount depends on current wire size and space. On the other hand, because of the nature of distributed RC network, the delay on upstream segments of e will also be impacted by any of the changes. So, the total changes on timing need to be evaluated on all impacted paths and sinks.

3.2 Capacitance transportation

We use ***sink(n)*** to denote the sink set for net ***n*** and ***path(e)*** to describe the path from source to segment ***e*** (not including e). When the capacitance on ***e*** has a change Δc , this change will be transported upstream to source and all segments in path(e) will have delay changes. We call this phenomena as ***capacitance transportation***. Based on the delay calculations described in previous section, we have:

a. for driver, the changed delay is: $\Delta d(g) = R_d(g) \bullet \Delta c$

b. for any sink ***si*** in Te, the delay change is:

$$\Delta d(s_i) = \Delta d(g) + \Delta d(e) + \left(\sum_{i \in path(e)} r(e) \right) \bullet \Delta c$$

To improve distance for the net, the ***necessary condition*** is: $\Delta d(si) < 0$.

c. for other sinks, the delay change (not include driver delay change) on sink i is:

$$\left(\sum_{j \in path(i) \cap path(e)} r(j) \right) \bullet \Delta c$$

With this calculation, we can quickly evaluate the impact on the total cost function and decide if this change is good or not. The capacitance change on ***e*** is always from wire sizing or wire spacing.

3.3 Some properties

For a segment e with current solution (w1, p1), we would like to study the possible sizing and spacing directions. For candidate solution (w2,p2) with w2= w1+ Δ and p2=p1+ Δ , we have three possible sizing and spacing paths, such as shown in Fig.4, where path1 = r1+r2; path2 = r3+r4; and path3 = r5.

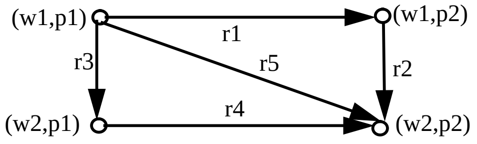

Fig. 4. Possible sizing and spacing directions.

Property 1: r1 and r4 always improve distance of the net.

Property 2: if r3 can not improve, then r2 can not improve either.

Property 3: improved r3 does not mean r2 can also improve.

Property 4: path2=r1+r2 is the safest path which has the best chance to achieve improvement.

The details about these properties will not be explained here. But the algorithm developed later will use the principles.

4. Algorithms

4.1 Section Constraint Graph

For each section, we can construct a constraint graph to represent the relationship between adjacent operating segments. We define ***section constraint graph*** (SCG) G=(V,E), where V is the set of vertices and E is the set of edges. Two special vertices vt and vb represent the top and bottom boundaries of the section. Any other vertex vi corresponds to horizontal operating segment i. A directed edge e(i,j) represents segment i and j are adjacent and i is above j. There are edges from vt to all top level vertices and from all bottom level vertices to vb. It should be noticed that it is not possible to have loops in a SCG because the routing is complete. Fig. 5(a) shows a section and its routing. Fig. 5(b) shows its initial constraint graph.

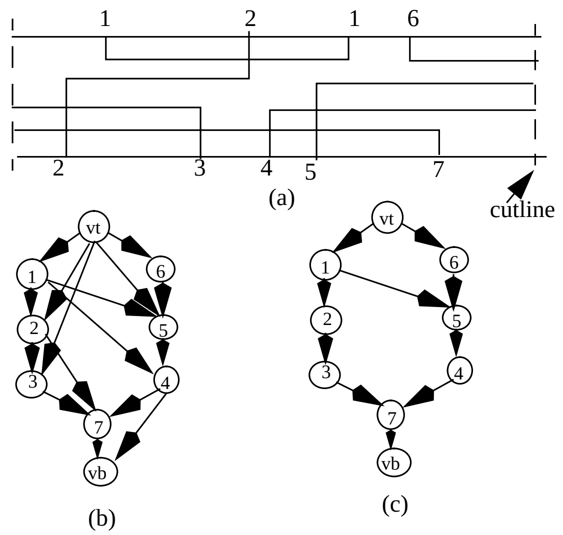

Fig. 5. Section Constraint Graph (SCG).
(a) Routing in a section. (b) Initial SCG.
(c) SCG after edge reduction.

Each vertex vi has a weight w(i) to represent the size of its corresponding segment and each edge e(i,j) has a weight p(i,j) to represent the space between i and j. ***Section height*** is defined as the sum of weights for all nodes and edges in the longest path from vt to vb. The nodes and edges on the longest paths are called ***SCG critical***. ***Channel height*** is defined as the maximum section height for all sections in the channel. The ***space slack*** of a section is the difference between channel height and section height. A ***critical section*** is the one having zero space slack. In the initial SCG, some edges may not be really useful because their weights will be implicitly decided by other edges (since segments in a section have relative short length, we do not consider the possible detour compaction case), such as edges e(1, 4) and e(2,7) in Fig. 5(b). Fig. 6 describes the algorithms for building SCG and reducing edges. Fig. 5(c) is the result after edge reduction.

```
scg_build:
  /* build initial SCG */
  for each track from bottom up;
    use projection for segments to extract edges;
    (each segment corresponding to a node)
    assign min. size and min.spacing for nodes and edges;
    update_scg(vt, downstream);   /* calculate path length from vt */
    update_scg(vb, upstream);     /* calculate path length from vb */

edge_reduction:
  /* remove useless edges */
  for each edge
    if (the difference between the two nodes' pathlength larger than
        max. spacing bound in WSSP)
        remove this edge;
```

```
update_scg(v, direction):
  /* update path information in SCG */
  use v as the starting node, navigate edges and nodes along given
  direction to update path lengths.
```

Fig. 6. Algorithms to build SCG.

4.2 Algorithm for WSSP

As described in section 2.4, WSSP can be formulated as a constraint-optimization problem. Considering the impact of wire sizing and wire spacing to the cost function, intuitively we think this problem is very complex and could be a NP issue (though we have no conclusion at this moment, we view the problem similar with 2D compaction and placement problems in some degree.) Based on this assumption, we develop a heuristic algorithm named as ***wssp***, which is described in Fig. 7. The algorithm consists of two phases with one phase for fixed area and the other phase for dynamically adjusted area. Since the detailed routing usually route all nets with minimum width and spacing rules, the initial channel height can be calculated first. With this height as fixed, we conduct phase one to do wire sizing and wire spacing to improve cost function (actually we do not need to calculate cost function, only the cost changes need to be measured.) After phase one, we try to select some channels to increase their height and evaluate if we can further improve the cost function. The algorithm stops once there is no way to improve or there is no illegal net left.

We start each segment and each edge in SCG from minimum width and minimum spacing and gradually increase them by Δ . A legal net will not be handled during sizing and spacing process. Since only wire spacing has impact to other adjacent nets and increasing spacing only improve performance, we know that a legal net will always be legal in the algorithm. With the progress of improvement, few illegal nets left and the algorithm converges to the solution.

```
wssp:
  build SCG for all sections and  calculate channel height;
  /* phase 1 - fixed all channel height */
  basic_wssp();
  /* phase 2 - dynamically adjust channel height */
  order all channels according to the number of critical sections;
  do
     for each channel
        increase channel size by $\Delta$;
        basic_wssp();
        if (there is no improvement)
           do not accept the change;
  until (there is no channel can be increased to improve cost)
basic_wssp:
   do
     edge_reduction for all sections;
     calculate distance for all critical nets;
     order illegal nets by their distance ;
     basic_iteration();
   until (there is no improvement in this iteration)
basic_iteration:
  for each illegal net
     for each segment e ( along top-down and BFS order)
        if (the section containing e is critical)
           if (e is SCG critical)
              /*  find all non-critical edges for e to do spacing */
              spacing (e);
           else
              /* do spacing followed by sizing */
              spacing (e);
              sizing(e);
        else
              get space slack in the section;
              spacing (e);
              sizing (e);
sizing (e):
  calculate how many $\Delta$ steps can be applied to e;
  (considering size bound,section height,space slack)
  increase size on e by the calculated $\Delta$ steps;
  if the distance of the net can be improved
     accept the sizing;
     update_scg();
spacing(e):
   /* handle incoming edges of e */
   for each incoming edge j of e
      if j is SCG critical and there is no space slack in the section
         skip j;
      else
         calculate how many $\Delta$ steps can be applied to j;
         (considering spacing bound,section height,space slack)
         increase space on j by the calculated $\Delta$ steps;
   update_scg();
   /* also handle outgoing edges of e with similar way */
```

Fig. 7. Algorithm description for WSSP.

4.3 Algorithm analysis

The algorithm described above iteratively improve cost function until there is no way to improve. Since legal nets are excluded in the iterations and it is not possible to have a legal net becoming illegal in the following iterations, illegal nets will gradually move toward legal status and the algorithm can always converge to a solution.

The complexities of the algorithm consists of:

sizing(e)/spacing(e):

the segments in path(e) need to be re-calculated. For a net with n segments, the average worst case time complexity is O(n) (if we apply sizing or spacing to each segment once, the worst case complexity is O(n^2)).

update_scg:

for graph G=(V,E), the worst case complexity to update it is O($|V| \bullet |E|$).

basic_iteration:

the maximum number of call to sizing or spacing functions is $2 \bullet |n_{il}| \bullet |os_h|$, where n_{il} is the number of illegal nets and os_h is the number of horizontal operating segments ($os_h < n$). So, the worst case complexity is O($|n_{il}| \bullet |V| \bullet |E| \bullet n^2$).

5. Preliminary Experiments

We have implemented WSSP algorithm in ANSI C for SUN SPARC station environment. In our experiments, to simplify problem, we assume all channels are horizontal and all horizontal segments are on the same layer. Also, vias are simply ignored in delay calculation. The basic parameters we used in the experiments are: r0=0.112 Ω /um, c0=0.039fF/um,Rd(g)=270.0 Ω ,pin load = 1.0fF. The minimum size and spacing are 0.5um and the maximum size and spacing are 1.5um. The Δ step is 0.25um. These parameters only serve as test purpose and may not be accurate in real world.

Three examples are used in testing WSSP. Since there is no existing benchmarks, here we randomly created these testcases. Table 1 gives the basic data about the three cases and Table 2 gives the experimental results. As we can see, after WSSP, the number of illegal nets and the average distance for illegal nets have been improved with very limited area penalty. In Table 2, area is measured by the sum of all channel heights

Since wire sizing and wire spacing techniques have its own limits, such as effective size and space bound, the area and timing trade-off, etc., we can not always get results without illegal nets. But the number of illegal nets and their average distance can be improved by the techniques.

Table 1: Basic data of testcases

Data	# of channels	# of tracks/ channel	# of nets	# of illegal nets	# of cut lines
Test1	2	2	6	4	1
Test2	3	3	8	5	1
Test3	4	3	12	6	2

Table 2: Experimental results

Data	Area		# of illegal nets		Avg. dist. of illegal nets	
	init.l	wssp	init.	wssp	initial	wssp
Test1	5.0	5.5	4	0	-402.5	0
Test2	10.5	2.0	5	1	-377.6	-98.7
Test3	14.0	5.0	6	1	-433.5	-101.8

6. Conclusions and future work

While wire sizing and wire spacing techniques can be used to improve interconnect delay caused by both ground and coupled capacitance, routing area also need to be considered. The proposed methodologies and algorithms for wire sizing and wire spacing try to trade-off both performance and area requirements. By partitioning routing regions into sections and constructing section constraint graphs, we developed a heuristic algorithm ***wssp*** to solve the wire sizing and wire spacing problem simultaneously. Preliminary results showed the efficiency of the techniques.

As feature sizes get into very deep-submicron domain and system-on-chip complexities get into hundred-million transistors, the on-chip performance, cross-talk, noise, power, and die-size issues are becoming more and more critical for success. All these issues are strongly related with interconnections. So, in our future work, these factors will be incorporated into wire sizing and wire spacing techniques to achieve more improvements

References

[1] Jason Cong, and Kwok-Shing Leung, "Optimal Wiresizing Under the Distributed Elmore Delay Model," Proc. ACM/IEEE Intl. Conf. Computer-Aided Design, pp.634-639, 1993.

[2] K.D.Bose, A.B.Kahng, B.A.McCoy, and G.Robins, "Rectilinear Steiner Trees with Minimum Elmore Delay," Proc. ACM/IEEE Design Automation Conf., pp.381-386, 1994.

[3] Chung-Ping Chen, Yao-Ping Chen, D.F.Wong, "Optimal Wire-Sizing Formula Under the Elmore Delay Model," Proc. ACM/IEEE Design Automation Conf., pp.487-490, 1996.

[4] Tianxiong Xue, and E.S.Kuh, "Post Routing Performance Optimization via Multi-Link Insertion and Non-Uniform Wiresizing," Proc. ACM/IEEE Intl. Conf. Computer-Aided Design, pp.575-580, 1995.

[5] J.L.Lillis, C.K.Cheng, and T.T.Y.Lin, "Optimal Wire Sizing and Buffer Insertion for Low Power and a Generalized Delay Model," Proc. ACM/IEEE Intl. Conf. Computer-Aided Design, pp.138-143, 1995.

[6] Qing Zhu, Wayne W.M. Dai, Joe G.Xi, "Optimal Sizing of High-Speed Clock Networks Based on Distributed RC and Lossy Transmission Line Models," Proc. ACM/IEEE Intl. Conf. Computer-Aided Design, pp.628-633, 1993.

[7] Jason Cong, Lei He, "Optimal Wiresizing for Interconnects with Multiple Sources," Proc. ACM/IEEE Intl. conf. Computer-Aided Design, pp.568-574, 1995.

[8] Kamal Chaudhary, Akira Onozawa, and E.S.Kuh, "A Spacing Algorithm for Performance Enhancement and Cross-talk Reduction," Proc. ACM/IEEE Intl. Conf. Computer-Aided Design, pp.697-702, 1993.

[9] D.A.Kirkpatrick, A.L.Sangiovanni-Vincentelli, "Techniques for Crosstalk Avoidance in the Physical Design of High-Performance Digital Systems," Proc. ACM/IEEE Int. Conf. Computer-Aided Design, pp.616-619, 1994.

[10] Tong Gao, C.L.Liu, "Minimum Crosstalk Channel Routing," Proc. ACM/IEEE Int. Conf. Computer-Aided Design, pp.692-696, 1993.

[11] W.C.Elmore, "The Transient Response of Damped Linear Networks with particular Regard to Wideband Amplifiers," J. Applied Physics, pp.55-63, 1948.

[12] H.B.Bakoglu, Circuits, Interconnections, and Packaging for VLSI, Addison-Wesley Publishing Company, 1990.

[13] T.Sakurai, K.Tamaru, "Simple Formulas for Two- and Three-Dimentional Capacitances," IEEE Trans. on Electron Devices, Vol.ED-30, pp.183-185, Feb., 1983.