Post-Routing Timing Optimization with Routing Characterization

Chieh Changfan†‡, *Yu-Chin Hsu*‡, *and Fur-Shing Tsai*‡
†Computer Science Dept. University of California, Riverside
‡Avant! Corporation, 46871 Bayside Parkway, Fremont, CA 94538

Abstract

Wire delay estimation has been a problem in designs of Very Deep Submicron(VDSM) technologies with feature size under 0.25 um. Conventional back-annotation approach does not guarantee timing convergence due to different estimation techniques for pre-layout and post-layoout timing. To solve the problem, a tight integration of logic synthesis with placement have been proposed recently. However, timing discrepancies between pre-routing and post-routing still exists due to unpredictable routing and cross coupling effects. In this paper, a post-routing timing optimization algorithm is presented. Experimental results show that this algorithm provides better result after detail routing is completed.

1 Introduction

With VLSI design technology advancing to under $0.25\mu m$ feature size, the delay due to the parasitic of wire routing has become a non-ignoring factor in circuit delay estimation. Traditional back-annotation approach which relies on the feedback loop between synthesis and layout cannot solve the timing problem because wire delay cannot be accurately estimated during the synthesis stage. It generally takes many iterations between synthesis and layout, and it does not guarantee on timing convergence.

To solve the problem, [1, 2] uses floorplanning during synthesis to estimate those long interconnect wires. A novel wire-plan approach is proposed in [3] to plan wire routing during synthesis. Methodologies are presented in [4] to exploit placement information during logic optimization. By working directly on a placement, more accurate wire parasitic information than statistical wire load can be used for logic timing optimization. However, there is still considerable timing discrepancies between pre-routing and post-routing estimation in a VDSM design. Table 1 shows some industrial designs on the timing estimation at different stages of physical design. It has shown timing discrepancies between pre-routing and post-routing designs.

Design	Size			Initial Slack(ns)	
	#cell	#macro	#net	before routing	after routing
ckt1	39135	17	42637	-0.808	-0.905
ckt2	32357	1	32942	-0.023	-0.195
ckt3	21516	1	23713	-2.717	-3.200
ckt4	96653	57	103319	-2.032	-2.598
ckt5	125412	10	137241	-3.215	-2.643
ckt6	83729	8	94623	-1.972	-2.394
ckt7	153079	15	164286	-1.736	-2.364
ckt8	97463	10	100251	-1.947	-1.586

Table 1: Timing report at different stages

We study the reason for the timing discrepancy in this paper. A post-routing logic optimization based on routing characterization is presented. With characteristics of routing factors, we can approximate the routing effect during optimization, thus better optimization result can be obtained after routing is done. Experimental result shows that optimization with routing characterization provides good fidelity between the timing of pre-routing and post-routing designs.

2 Timing Fidelity Before and After Routing

Minimal rectilinear spanning tree or Steiner tree is usually used to estimate the wire load and delay for the interconnects of a placement. For designs with feature size $0.5um$ or larger, such an estimation is enough to achieve a fidelity between the timing of pre-routing and post-routing designs. However, in a VDSM design, routing congestion is more severe than ever, wires have to detour around the over-congested area and the coupling effect is usually large. These factors will result in timing discrepancies between pre-routing timing estimation and post-routing timing estimation. The timing fidelity between these two stages

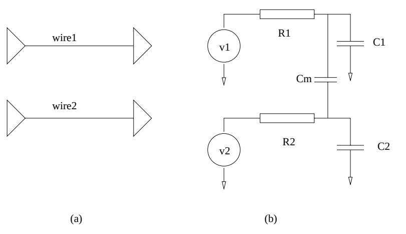


Figure 1: Coupling effect modeling

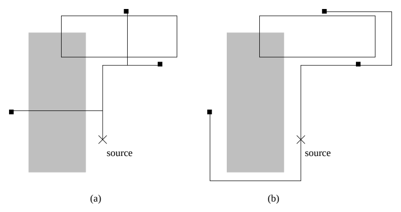


Figure 2: Routing pattern effect

is no longer guaranteed. Although there are other factors that may cause timing discrepancies between pre-route and post-route design, we observe that *routing pattern* and *coupling effect* are the major factors that cause timing discrepancies.

2.1 Coupling Effect

More and more devices are placed in a single chip in a VDSM design. Reduction in the interconnection wire and transistor switching delay results in faster signal transition time. All these factors increase the coupling effect between interconnection wires. Coupling not only increases signal delays, but also introduces noise over neighboring wires.

Coupling between two neighboring wires can be characterized by a simple model in Fig 1. In this model, we ignore the inductive coupling and consider only capacitive coupling between two wire segments. Let v_1, v_2 be the voltages at the outputs of the driver, R_1, R_2 be the wire resistances, C_1 and C_2 be the intrinsic capacitances of each wire respectively, and C_m be the coupling capacitance between wires.

In general, each element in a chip is coupled with every other element. Coupling capacitance decreases rapidly when an element is out of the neighborhood of the other element. Moreover, coupling capacitance between perpendicular wires is very small. Therefore, we assume coupling capacitance only exists between neighboring parallel wires. It can be estimated by

$$C = \alpha * \frac{length}{distance^{\beta}} \tag{1}$$

where *length* is the length of paralleled wire segments, *distance* is the distance between two wires, α and β are constants[5].

Smaller feature size means smaller wire width, which also means smaller intrinsic capacitance C_i of a wire. In the mean time, higher integration causes smaller distance between wires and longer average paralleled wires. These factors result in large coupling capacitance C_m between wires. In VDSM technology, coupling capacitance may even exceed the intrinsic capacitance and dominates the wire load.

2.2 Routing Pattern Effect

In timing analysis of a pre-routing design, the routing of a net is usually assumed to be a minimal spanning tree or a rectilinear Steiner tree, as shown in Fig 2(a). A minimal rectilinear Steiner tree represents a lower-bound in terms of total wire length. In this figure, the shaded area indicates routing congested area which is impossible for new wires to be added, and the boxes represent routing blockages. To avoid routing through the blockage areas, the final detailed routing may look like the one in Fig 2(b). The capacitance of this routing tree is larger than the one with minimal Steiner tree, therefore the actual delay along this net is larger than the delay estimated by a minimal Steiner tree. The more the routing congestion is, the larger the probability a physical routing of a net will detour, and the larger the timing discrepancy will be found after routing.

3 Routing Characterization

As described in the previous section, routing pattern and coupling effect are the major reasons of no timing fidelity between the timing estimation of pre-route and post-route designs. If the effect of these two factors can be predicted, the timing estimation of pre-routing design can be used for timing optimization. By analyzing the existing routing, we can characterize the coupling effect of the whole design, and predict the routing pattern.

3.1 Coupling Characterization

To characterize the coupling capacitance, the layout floorplan is first divided into 3D routing plane of small regions, each of which can be viewed as a single-layer rectangular area, as shown in Fig 3. Within each region r, the routing

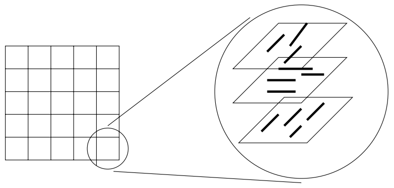


Figure 3: Coupling characterization

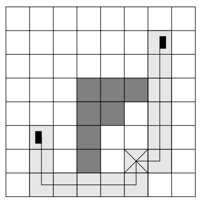


Figure 4: Routing prediction

congestion is defined as

$$\Omega(r) = N(r)/T(r),$$

where $N(r)$ is the number of nets in this region, and $T(r)$ is the routing capacity (number of tracks) of this region. Assume that wires in each region are distributed evenly, the expected distance between two wires can be estimated by $\bar{d}(r) = \delta/\Omega(r)$, where δ is the space between two tracks. If a wire goes through this region by length L, according to Eq. 1, the expected coupling capacitance is $(\alpha * L/\bar{d}(r)^{\beta})$, or equivalently $(\alpha * L * (\Omega/\delta)^{\beta}(r))$. We can characterize the unit-length expected coupling capacitance within region r as

$$C_l(r) = \alpha' * \Omega^{\beta}(r).$$

For each region, the unit-length expected coupling capacitance is characterized. The smaller the regions, the more accurate the coupling factor is characterized. With this characterization, we can estimate the coupling capacitance for a new net by the length within the regions it goes through.

3.2 Routing Pattern Prediction

Even with a good characterization of coupling effects, we still cannot obtain accurate coupling capacitance because the regions a net will go through are still unknown. A routing pattern prediction is required to predict which regions the final routing will go through. If there is blockage or over-congested area, we will use routing pattern prediction to estimate the possible routing at this stage. This prediction can be passed down to a detailed router to guide the final routing so that more accurate prediction at the placement level can be achieved.

The routing pattern prediction problem can be formulated as follows.

Problem 1. Routing Pattern Prediction Problem (RPP)
Given a routing graph $G = (V, E)$ with routing solutions for nets 1, ..., $m-1$, the problem is to find a set S of connected regions which cover all terminals of this net with the objective to

$$\text{Minimize} \sum_{r \in S} l(r) * C_l(r)$$

subject to capacity constraints

$$\Omega(r) \le \Omega_m(r) \text{ for each region } r,$$

where $l(r)$ is the expected length within region r, and $\Omega_m(r)$ is the maximum congestion allowed within region r. $\diamond$

Fig 4 shows an example of a region assignment for routing prediction. The X mark is the source of the net, and the black spots are the sinks. The dark-shaded regions are blockages or over-congested areas, and the light-shaded regions are selected regions. The complexity of a bruise force approach to all possible solution is $O(2^n)$, where n is the number of regions. Such an approach is not suitable for our application.

A weighted rectilinear Steiner tree heuristic [7] is adopted to our application. The cost of the edge connecting two neighboring regions r_1 and r_2 at the same layer is $(Cost(r_1) + Cost(r_2))/2$, where $Cost(r) = C_l(r)/(\Omega_m(r) - \Omega(r))$. Based on the cost function, the cost will approach to infinity when $\Omega(r)$ is close to the maximum congestion $\Omega_m(r)$. A pseudo code of this algorithm is shown in Fig 5.

4 Post-Routing Optimization

The post-routing logic optimization flow is shown in Fig 6. The proposed approach includes incremental optimization on a routed design by iterating the following steps: incremental timing analysis, cluster selection, logic transformation, incremental placement and routing characterization. The pseudo code of this algorithm is as shown in Fig 7.

```
Routing_Prediction(
   localized_window R,
   net N,
   placement PL,
   routing_characteristic RC         )
{
   T = Prim_MST(N,PL,R); /* localized within R */
   L[0] = empty;
   for j = 1 to number_of_edge(T)
   {
       min_cost=infinite;
       e[j] = an edge of T;
       P[j]= a subset of staircase layout of edge e[j];
       for i = 1 to sizeof(P[j])
       {
           Let P[j,i] be the i-th element of P[j];
           Q[j,i] = Merge(L[j-1], P[j,i]);
           cost = Weight(Q[j,i],RC);
           if (cost < min_cost)
           {
              L[j]= Q[j,i];
              min_cost = cost;
           }
       }
   }
}
```

Figure 5: Algorithm for routing prediction

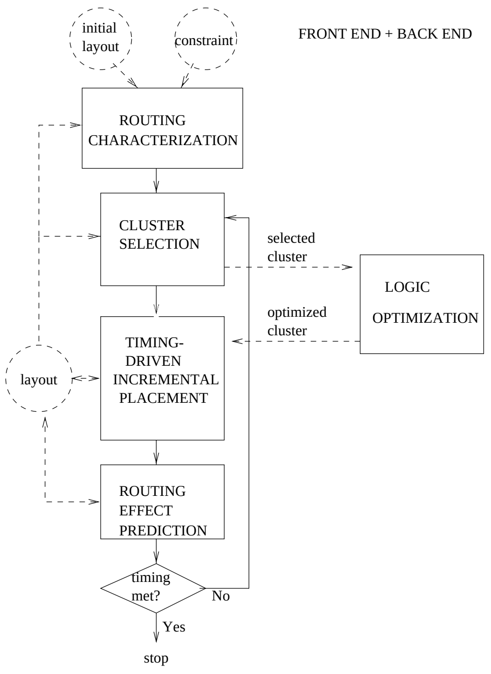


Figure 6: Post-routing timing optimization flow

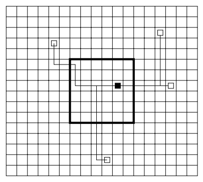
```
Post_Routing_Logic_Optimization(
                  netlist N,
                  library L,
                  timing_constraint T,
                  placement PL,
                  routing RT,
                  localization_window_size WS  )
{
    iteration = 0;
    SlackGraph = Timing_Analysis(N,L,T,PL,RT);
    RC = Characterize_routing(N,PL,RT);
    while ( timing_constraint_violated &&
            stop_criteria_not_reached_yet  )
    {
        /* cluster selection */
        s  = seed_selection(N,SlackGraph,PL,WS,iteration);
        W  = get_localization_window(s,WS);
        N0 = Cluster_grouping(s,N,SlackGraph,iteration,W);

        /* logic optimization */
        N1 = Logic_Optimization(N0,L,W);

        /* incremental placement */
        PL1 = TIP(N,L,T,PL,N0,N1,W);

        /* routing prediction */
        RT'=routing_prediction(W,N,PL1,RC);

        /* incremental timing analysis */
        SlackGraph1 = Incr_Timing_Analysis(N-N0+N1,L,T,PL1);
        if (critical_slack(SlackGraph1) >
                critical_slack(SlackGraph))
        {
           /* timing improved */
           N = N - N0 + N1;
           PL = PL1;          /* use new placement */
           SlackGraph = SlackGraph1;
        }
        /* else: give up current change */
        iteration++;
    }
}
```

Figure 7: Algorithm for post-routing optimization

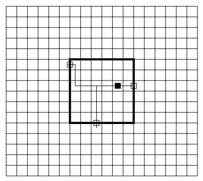
4.1 Cluster Selection

To select clusters to be optimized, the algorithm selects a set of gates around critical paths. Such selection can be divided into two steps: *seed selection* and *grouping*. In the seed selection step, candidate gates are selected from the ϵ-network, which is defined as a cluster where all signals have a slack in the range of $(Critical\_Slack, Critical\_Slack+\epsilon)$, where $Critical\_Slack$ is the critical slack of this design and ϵ is a constant. During the grouping step, the neighboring gates around each seed are selected to form a logic cluster. Since the optimization is done at the post routing stage, the cluster size is limited to very small so that the perturbance to the layout is minimal.

During the seed selection step, the set of candidate gates in the ϵ-network is selected at each iteration of incremental optimization. The selection is based on a cost function with four weighted criteria. The first criteria is the criticality of the gate. The second is the difference among the arrival times of the inputs to the gates, which indicates the potential of timing improvement by restructuring the gate. The third criteria is the number of fan-in's and fan-out's of the gates in the ϵ-network, which shows the potential timing influence of the gate on the logically adjacent gates. The fourth criteria is the congestion of the neighboring area. Instances in the congested area are not preferred be-



(a) Original placement and routing



(b) Add pseudo instances for localization

Figure 8: Localization of changes

cause changes within this area will cause larger disturbance on the routing characteristics.

After a seed is selected, the grouping process clusters the adjacent instances to the seed selected to form a partition. A user-specified window size is given to control the logic change within a localized area. A window centered at the seed instance is considered in grouping process. When grouping instances, we follow the logic connection of seed to find the instances to be grouped. If a instance is not within the window, it will not be selected even if it connects to the seed.

4.2 Incremental Placement

As soon as the new instances are generated by logic restructuring, they are placed back to the design within the window the old cluster is selected. To restrict the change within the window, when a wire crosses the boundary of the window, a pseudo instance connected to this wire will be created at the boundary of the window, as shown in Fig 8. The size of windows can be decided according to the congestion of the design.

An incremental placement algorithm presented is applied to solve this problem. This approach is divided into two steps: *timing-driven global placement* and *overlap removal*.

In the global placement phase, we try to find a placement solution for new instances which maximizes the critical slack with consideration of cell density. The objective function of our problem is

$$\text{Maximize } Min\{r_i - a_i; \text{ for each constrained port } i \text{ in top level}\} \tag{2}$$

where r_i is the required time at node i, and a_i is the arrival time.

In order to apply the first-order derivative to this objective function, we transform this objective function into an analytical form:

$$\text{Minimize} \sum_{i \in constrained\_ports} e^{-(r_i - a_i)}. \tag{3}$$

Here linear model is assumed. That is, the gate delay from input i to output j is $d(i,j) = u_1^{i,j} C_w^j + u_0^{i,j}$. By applying first-order partial derivative over the location x_k of instance k, we obtain

$$\sum_{i \in CP} e^{-(r_i - a_i)} \left(\sum_{(i,j) \in IP} u_1^{i,j} \frac{\partial C_w^j}{\partial x_k} \right) = 0 \tag{4}$$

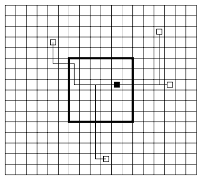
where CP is the set of constrained ports, IP is the set of internal delay edges(gate input to output) along the path, C_w^j is the output load at gate output j including wire load and gate capacitance. To speed up the computation, some insignificant terms are eliminated from our equation. Since for each location variable x_k, an equation is generated, there are $2n$ simultaneous equations which will decide the placement of these n instances.

The overlap removal algorithm removes overlap violation by moving these instances subject to their maximum move constraints while maximizing critical slack simultaneously. The maximum move constraint is given according to the local congestion and the criticality of an instance. localization constraints are applied to restrict the placement change within a localized window.

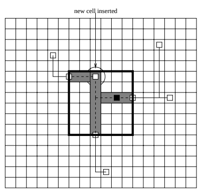
4.3 Routing Characterization

A quick routing tree prediction algorithm described in Sec. 3.2 is applied to the partially routed design. The coupling capacitance is estimated by the regions the routing tree goes through. For nets extended to outside of the window, there must exist a corresponding pseudo instance at the window boundary. When estimating the routing, we only connect this net to the pseudo instance instead of connecting it outside the window. In this way, the routing outside the window will not be changed. An example of localized routing prediction is shown in Fig. 9.

After the routing prediction is done, timing analyzer may use the coupling capacitance to estimate the timing



(a) Routing before optimization



(b) Routing prediction after optimization

Figure 9: Routing prediction strategy

Design routing	Pre-routing Opt(ns)		Post-routing Opt(ns)	
	before	after	before	after
ckt1	-0.869	-0.934	-0.755	-0.775
ckt2	-0.143	-0.221	-0.175	-0.186
ckt3	-3.099	-3.175	-3.167	-3.186
ckt4	-1.478	-4.407	-1.396	-1.830
ckt5	-1.908	-3.012	-1.894	-1.925
ckt6	-1.424	-2.144	-1.521	-1.642
ckt7	-1.312	-1.986	-1.324	-1.335
ckt8	+0.012	-0.921	-0.354	-0.435

*

design size and initial slack are shown in Table 1.

Table 2: Experimental Results

after routing. If it improves the timing, this change will be committed, and the routing characteristics will be updated according the global routing tree. The global routing tree will also be recorded so that it can be used to guide the detailed routing of the optimized netlist.

5 Experimental Results

The post-routing timing optimization algorithm proposed in this paper has been implemented and tested on the cases shown in Table 1. The result is shown in Table 2.

The initial design has been optimized by the algorithm presented in [4] and routed by commercial routing tools. The algorithm in [4] is applied to the routed design again, and the result is shown in the column `Pre-routing Opt`. The algorithm presented in this paper is also applied to the same design, and result is shown in the column `Post-routing Opt`. As shown in this table, pre-routing optimization algorithm fails to hold fidelity after routing. The post-routing optimization algorithm gives a good fidelity and thus further improves timing after routing.

6 Conclusions

In this paper, we study the timing correlation problem between pre-route and post-route designs. Coupling effect and routing pattern effect are identified to be the two major factors that cause the timing discrepancies. We propose a post-routing timing optimization algorithm based on routing characterization. Experimental result shows that algorithm considering coupling effect and routing pattern preserve good correlation after ECO routing.

References

[1] H.P. Su, A. Wu, Y.L. Lin, "Performance-Driven Soft-Macro Clustering and Placement by Preserving HDL Design Hierarchy", *Proc ISPD* 1998, pp. 12-17.

[2] A.H. Salek, J. Lou and M. Pedram, "A DSM Design Flow: Putting Floorplanning, Technology-Mapping, and Gate-Placement Together", *Proc DAC* 1998, pp.128-133.

[3] R. H.J.M. Otten and R. K. Brayton, "Planning for Performance", *Proc DAC* 1998, pp.122-127.

[4] M. T-C. Lee, T-Y. Liu, K. F-S. Tsai, E. S-Z. Lin, and Y. C. Hsu, "Incremental Timing Optimization for Physical Design by Interacting Logic Restructuring and Layout," *International Workshop on Logic Synthesis*, June, 1998.

[5] T. Sakurai and K. Tamaru, "Simple formulas for two and three dimensional capacitance," *IEEE Trans. Electronic Devices,* 1993.

[6] H. Zhou and D.F. Wong, "Global Routing with Crosstalk Constraints," *Proc DAC* 1998, pp.374-377.

[7] C. Chiang, M. Sarrafzadeh, and C.K. Wong, "A powerful global router: Based on Steiner min-max tree." *Proc DAC* 1989, pp.2-5.

[8] J. Lee and D.T. Tang, "An algorithm for incremental timing analysis" *Proc DAC 1995*, pp.696-701.

[9] K.J. Singh, A.R. Wang, R.K. Brayton, and A.Sangiovanni-Vincentelli, "Timing Optimization of Combinational Logic," *Proc ICCAD* 1988, pp282-285.

[10] R.L. Rudell, *Logic Synthesis for VLSI Design.* PhD thesis, Electronics Research Lab., Univ. of California, Berkeley, Apr. 1989.