Post-Layout Logic Restructuring for Performance Optimization

Yi-Min Jiang  Angela Krstic  Kwang-Ting Cheng  Malgorzata Marek-Sadowska

Department of Electrical & Computer Engineering
University of California, Santa Barbara, CA 93106

Abstract

We propose a new methodology based on incremental logic restructuring for post-layout performance improvement. The new post-layout logic restructuring technique allows to use accurate interconnection delays for performance optimization, while the incremental nature of the technique guarantees convergence between logic synthesis and layout. The technique can be further integrated with other post-layout optimization techniques such as gate sizing and buffer insertion. Experimental results show that this technique combined with post-layout buffer insertion can achieve an additional 15% improvement in performance compared to designs produced by timing-driven logic optimization followed by pre-layout buffer insertion followed by timing-driven physical design.

1. Introduction

Performance-driven logic synthesis followed by performance-driven layout [1] has become a necessity for designing high performance circuits. However, this loosely coupled two-phase timing optimization methodology has serious limitations for deep-submicron-based designs in which interconnection delays become the dominant factor in determining the circuit speed. Accurate information regarding the interconnection delays is not available during the logic synthesis phase and the interconnection delays are just roughly estimated in this phase. Therefore, errors in the estimation could result in a logic design far-off from an optimal one.

Recently, an ATPG-based approach, named Redundancy Addition and Removal, was proposed for combinational and synchronous sequential logic optimization [3][4][5]. This approach optimizes the networks through iterative addition and removal of redundant connections. The redundancy addition and removal technique can identify alternative connections or gates for any given connection. For performance optimization, we can remove a connection on a critical path and replace it by one of its alternative connections/gates not on the critical path. Since the layout of the design is available, routing for each alternative connection can be attempted and its interconnection delay can be accurately calculated. Therefore, good alternative connections/gates for replacement can be found in a greedy manner and the replacement can always achieve a real improvement of the circuit performance.

In this paper, we apply the redundancy addition and removal technique for post-layout performance optimization. We further combine the technique with buffer insertion. The proposed method can be used to rectify designs produced by any synthesis tool that is driven by inaccurate cost functions. It allows the use of accurate interconnection delays for performance optimization and its incremental nature guarantees that the logic transformation can be physically implemented without changing the layout of the rest of the logic.

We built a prototype system to implement the post-layout optimization process. The system consists of the Berkeley synthesis system SIS 1.2 [6], a commercial physical design system GARDS [7], and our programs for logic restructuring and for buffer tree insertion. The system has been tested for a suite of benchmark circuits. For each benchmark circuit, we generate a performance-optimized design using SIS 1.2 and GARDS. We then apply post-layout performance optimization using three different strategies: (1) using post-layout buffer insertion only, (2) using post-layout logic restructuring only and (3) using both logic restructuring and buffer insertion. For the 28 circuits we tested, we have observed, on average, an improvement of 6.6%, 7.6% and 15.4% respectively for these three strategies. Details of the experiments and the results are given in Section 3.

This paper is organized as follows. In Section 2 we discuss how to apply this technique for post-layout performance optimization and how to combine it with the buffer insertion technique. Section 3 gives the details of implementation and experimental results on ISCAS89 and MCNC benchmark circuits. Section 4 concludes the paper.

2. Post-layout Performance Optimization

Our proposed post-layout optimization process is shown in Figure 1. Initially, the logic is optimized and the buffers are inserted for performance optimization using the floorplan information. It is then followed by the performance-driven placement and routing. After the physical design phase, the interconnection delays are extracted and calculated, and timing-critical wires (wires on the critical paths) are identified using a static timing analyzer. A timing-criti-
A design may have a large number of timing-critical paths. We need to develop a strategy for determining which nets to target and their order for processing and removal. (2) A given target wire might have many alternatives. The selection of an alternative wire for replacement of the given wire can also substantially affect the final results. In the following we discuss these problems in more detail and describe our solutions to them.

### 2.1 Selecting timing-critical wires for removal and alternative wires for addition

A wire \( w = g_i \rightarrow g_j \) is called a *candidate wire* if it is on a timing-critical path of the circuit. We say that logic \( l \) consisting of one or more wires and/or gates is an *alternative logic* for wire \( w \) if the addition of logic \( l \), together with the removal of wire \( w \), does not change the function of any of the outputs of the circuit. A pair \((w, l)\) is said to be a *candidate pair* if \( w \) is a candidate wire and \( l \) is an alternative logic for \( w \). We consider the path with the longest delay as a timing-critical path.

We use a static timing analyzer to identify a set of timing-critical paths. A subset of wires contained in the critical paths are considered as candidate wires. Each candidate wire may have more than one alternative logic. Therefore, there is a set of candidate pairs. Our goal is to identify a candidate pair among the set such that, after the transformation is performed, the performance of the circuit improves the most. Adding and removing redundancy changes the logic structure and the timing information in the circuit. Let the candidate wire for removal be \( w_r = g_i \rightarrow g_j \) and let wire \( w_a = g_m \rightarrow g_n \) be its alternative wire as shown in Figure 2.

The fanout cones of these four gates are as indicated.

![Figure 2: Regions of signals affected by redundancy addition and removal](image)

Removal of the candidate wire results in the removal of the target critical path and a number of other paths. The load of gate \( g_i \) decreases and therefore, the arrival times of gates in the fanout cone of gate \( g_i \) may change (i.e., the delays of all remaining paths that contain gate \( g_i \) decrease). On the other hand, addition of the alternative wire \( g_m \rightarrow g_n \) results in a number of paths being added into the circuit. The logic function of all internal gates in the fanout cone of the gate \( g_n \) may change. The load of the gate \( g_m \) increases and that affects the arrival times of the gates in the fanout cone of the gate \( g_m \) The discussion can be easily generalized for cases when gates, instead of wires, are added.

Since in the transformation selection phase the alternative connections/gates are not physically implemented, we estimate the delay of the added logic based on the current
layout information. Once an estimated delay of the alternative wire is derived, incremental timing analysis which examines only the nodes in the fanout cones of the associated gates can derive an estimated delay of the circuit after the transformation.

The alternative wire/logic may need to be added in a congested area where it cannot not be properly placed and routed. A parameter indicating the degree of congestion of a target area can be directly derived by the physical design tool we use [7]. We set a threshold value to exclude some candidate transformations. If the congestion parameter of the area of the alternative wire/logic is greater than the threshold, we remove the corresponding candidate transformation from the candidate list.

Modifying the circuit based on a candidate pair may affect many other paths in addition to the targeted timing-critical path. The impact on other long paths should also be considered in the cost function for selecting the best candidate pair. We define the benefit function of a candidate pair \((w,l)\) as follows:

\[
\text{benefit}(w,l) = \text{gain}(w,l) + \alpha \times \text{secondary\_gain}(w,l) \quad (1)
\]

The gain of a candidate pair \((w,l)\) is defined as the reduction of the circuit delay obtained by replacing the candidate wire \(w\) with its alternative logic \(l\). The secondary_gain of \((w,l)\) is the delay reduction of the \(n\) longest paths by the transformation where \(n\) is a program parameter. A secondary_gain\((w,l)\) which reflects the delay reductions with respect to a subset of long paths gives guidance for selection. In our experiment, \(n\) is chosen as 5. The weighting constant \(\alpha\) is a small number which is assigned as 0.1 in our experiment. The benefit function is computed for each candidate pair and the one with the highest value is selected for implementation.

### 2.2 Combining redundancy addition and removal with buffer insertion

In the transformation selection phase, in addition to considering the candidate pairs, we can also consider inserting buffer trees at the timing-critical nets or sizing their driving gates. These three types of transformations are significantly different in nature. Therefore, combining them would allow exploring a larger design space and lead to better results - this could particularly be true under the greedy strategy used in our iterative process. In this paper, we choose to include buffer insertion only to illustrate that multiple types of transformations can be easily integrated into our post-layout optimization process.

For each timing-critical net, we consider inserting a \(k\)-ary balance tree [2] - the number of fanouts of each node in the tree is \(k\) except for the leaf nodes. We use the delay equation proposed in [2] to estimate the delay from the source node to each of the leaf nodes after the buffer tree is inserted. If a buffer insertion transformation has the highest benefit among all candidate transformations considered, we construct the buffer tree for final implementation.

### 3. Implementation and Experimental Results

In this section, we discuss some implementation issues and present experimental results for ISCAS89 and MCNC benchmark circuits.

We use the SIS-1.2 package [6] for logic optimization and GARDS [7] for layout. For logic optimization, script script_delay is applied followed by pre-layout buffer insertion buffer_opt to optimize the circuit for performance at the logic level. For physical design, we use a 0.55 \(\mu\)m CMOS library and GARDS [20] - a physical design system for gate arrays design. We use the performance-driven option in GARDS for performance-driven placement and routing. After physical design, we further use GARDS to extract the RC-trees of the interconnects and calculate the interconnect delays for timing analysis.

For each circuit, we follow the algorithm shown in Figure 1 for post-layout performance optimization. In order to evaluate the redundancy addition and removal technique and the value of combining it with the buffer insertion technique for post-layout optimization, three different strategies are used: (1) buffer insertion only, (2) redundancy addition and removal only and (3) combining both buffer insertion and redundancy addition and removal.

Once a post-layout transformation is identified (either replacing a wire with its alternative logic or inserting a buffer tree to a net), we use the Engineering Change Orders (ECO’s) feature of GARDS to implement the transformation. It tries to keep the layout of the unmodified logic intact, and only re-place and re-route the modified part of the circuit. The router used in GARDS is an area router which makes incremental change of routing possible.

The results for the 12 largest ISCAS89 sequential benchmark circuits and 14 largest MCNC combinational benchmark circuits are summarized in Tables 1 and 2. The delays of the longest paths for circuits optimized by SIS with script.delay and buffer_opt followed by GARDS for timing-driven layout are shown in Column 2. The interconnect delays for timing analysis are computed by GARDS using the extracted RC-trees. Columns 3-6, 7-10 and 11-14 shows the results of (1) using buffer insertion alone, (2) using redundancy addition and removal alone and (3) using the combination of buffer insertion and redundancy addition and removal. For each experiment, we record the critical path delay of the final design, the percentage of reduction of critical path delay, the number of iterations and the total runtime of post-layout optimization.

Post-layout buffer insertion reduces the critical path delays, on average, by 5.8% and 7.4% respectively for the two sets of benchmark circuits. Redundancy addition and removal reduces them by 7.6% and 7.6% respectively which is just slightly better than the buffer insertion technique. When these two techniques are combined, 15.3% and 15.4% reductions are achieved which are even greater than the sum of the reduction achieved by each individual technique. These results indicate that these two post-layout...
transformations are complementing each other. The results also indicate that the redundancy addition and removal tech-
nique and the combined technique require more iterations
and therefore take longer CPU runtimes. The CPU runtimes
shown in the tables include the runtimes for all post-layout
processes including incremental layout (through ECO of
GARDS), timing analysis and logic transformation identifi-
ation. We have demonstrated
the effectiveness of the technique. We have demonstrated
that the technique is complementary to the buffer insertion
technique. By combining buffer insertion and the new logic
restructuring technique, another 15% improvement on per-
formance can be achieved for designs already timing-opti-
mized by synthesis and layout tools. We are currently
investigating more logic transformations which are suitable
in the post-layout phase.

### References


struction Heuristics for the Fanout Problem,” Proceedings of
International Conference on Computer-Aided Design, pp. 671-

by Redundancy Addition and Removal,” Proceedings of Euro-
pean Design Automation Conference, pp. 373-377, February
1995.

by Redundancy Addition and Removal,” Proceedings of Euro-
pean Design Automation Conference, pp. 373-377, February
1995.

and Simplify: Multi-level Boolean Network Optimizer,” IEEE Transactions
on CAD, vol. 15, no. 12, pp. 1494-1504, December
1996.
