Multilevel Global Placement with Retiming

Jason Cong
Department of Computer Science
University of California, Los Angeles, CA 90095
cong@cs.ucla.edu

Xin Yuan
Department of Computer Science
University of California, Los Angeles, CA 90095
yuanxin@cs.ucla.edu

ABSTRACT

Multiple clock cycles are needed to cross the global interconnects for multi-gigahertz designs in nanometer technologies. For synchronous designs, this requires retiming and pipelining on global interconnects. In this paper, we present a practical solution for simultaneous retiming and multilevel global placement for performance optimization, based on the theory and algorithms of *sequential timing analysis* (Seq-TA). We extend the Seq-TA to handle gates/clusters with multiple outputs and integrate it into a multilevel optimization framework for simultaneous retiming and placement. We also develop two speed-up techniques which enable the Seq-TA to be efficiently integrated into a simulated annealing-based multilevel coarse placement for large-scale designs. Experimental results show that (i) retiming can improve the performance (delay) by 14% on average when it is applied after placement; (ii) our approach for simultaneous retiming and placement can outperform the two-step approach (placement followed by retiming) by 10% on average in terms of delay minimization.

Categories and Subject Descriptors

B.7.2 [**Hardware**]: INTEGRATED CIRCUITS—*Design Aids*

General Terms

Algorithms, Design, Experimentation, Performance

Keywords

Placement, retiming, physical hierarchy, deep sub-micron

1. INTRODUCTION

The International Technology Roadmap for Semiconductors (ITRS'2002 update) [18] predicts that there will be over ten billion transistors integrated on a single chip with an on-chip local clock frequency of 28GHz in the 22nm technology by 2016. It was shown in [3] that even with the use of new interconnect materials and aggressive interconnect optimization, the delay of a 2cm global interconnect still remains around 500ps. This implies that multi-cycle communications over the long interconnects are required for multi-gigahertz synchronous designs.

Retiming is a powerful sequential optimization technique used to minimize the clock period (delay) or the number of flipflops (FFs) by relocating the FFs (it changes the netlists) while preserving the functionality of the circuits [11]. The potential of retiming over the global interconnects needs to be considered during the global placement stage, as the placement results define the interconnects. Given a netlist of the circuit, the existing placement algorithms place the gates and blocks so that certain objectives, e.g., wire-length minimization, delay minimization, and routing congestion minimization, can be achieved. They can not change the netlist of the circuit. However, the benefit of considering retiming during the placement stage is significant and can be easily illustrated by the simple motivational example shown in Figure 1. A circuit G shown in Figure 1(a) consists of two PIs, I_1, I_2, two POs, O_1, O_2, four gates, a, b, c, d, and one flipflop F_1. We assume that each gate has delay of 1, and the interconnect delay between a PI (PO) and a gate (FF) is zero. The interconnect delay between two gates (FFs) is marked on the edge connecting them in the figure. Given Placement 0 of G (shown in Figure 1(a)) with delay of 5, a timing-driven placer will identify the critical path based on the static timing analysis and try to minimize the longest path, thus generate Placement 1 (shown in Figure 1(b)) with delay of 4. After delay minimization retiming is performed on Placement 1 (shown in Figure 1(c)) by moving flipflop F_1 from the fanin of gate a to its fanouts, the longest path delay is unchanged. On the other hand, if the placer is aware of retiming possibility along the path from $I_1 \to F_1 \to a \to d \to O_2$, it will identify critical paths in terms of retiming potential and try to short them, thus generate Placement 2 (shown in Figure 1(d)) with path delay of 5. However, after retiming is applied on Placement 2 (shown in Figure 1(e)), the delay can be further reduced to 3. This shows the necessity of considering retiming during the placement stage in order to hide long interconnect latency for performance optimization, because not all the long interconnects are problematic. Only those long interconnects which have to be crossed in a single clock cycle are problematic and should be optimized during the placement stage. In the above example, path $p_r = I_2 \to c \to d \to O_2$ has to be crossed in a single clock cycle, while path $p = I_1 \to F_1 \to a \to d \to O_2$ can be crossed in two clock cycles if retiming is applied. Therefore path p_r is a "bad" interconnect, while path p is not. Although Placement 1 is better than Placement 2 in terms of delay before retiming is performed, Placement 2 is better than Placement 1 in terms of retiming possibility as it leads to better performance after retiming. The critical path p_r ("bad" interconnects) in the retimed circuit is shortened in Placement 2, while the apparent critical path

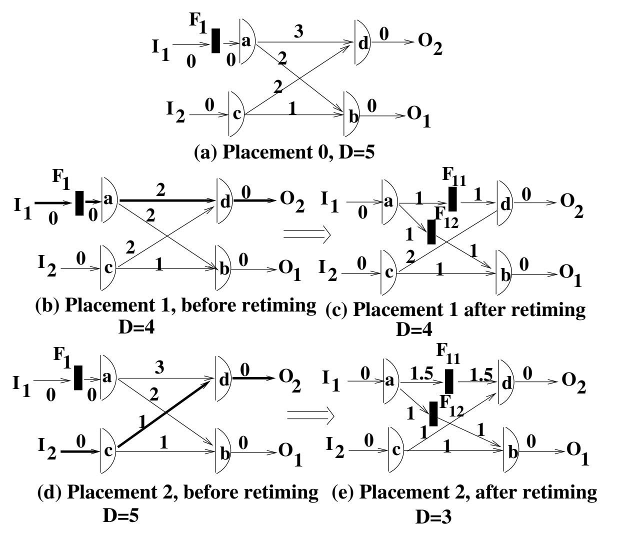
(a) Placement 0, D=5
(b) Placement 1, before retiming D=4
(c) Placement 1 after retiming D=4
(d) Placement 2, before retiming D=5
(e) Placement 2, after retiming D=3
F1
F11
F12
I1
I2
O1
O2
a
b
c
d

Figure 1: Advantage of simultaneous placement and retiming for delay minimization

p in the pre-retimed circuit (regarded as "bad" interconnects by the traditional placers) is shortened in Placement 1.

There are two kinds of approaches, iterative and simultaneous, for integrating retiming with placement or floorplanning in the physical design phase. In the iterative approach [21, 13, 14], placement and retiming are alternatively performed until the timing constraints are met, that is, wirelength-driven or timing-driven placement is first performed on the given netlist to minimize the total wirelength and/or delay followed by performing retiming on the placed circuit based on the layout information. In the simultaneous approach [5, 2, 20], placement or floorplanning is performed in such a way that the placement or floorplanning engine can be aware of the retiming possibility and thus reduce the lengths of interconnects which are critical in terms of retiming potentials. In the simultaneous approach, the placer with retiming awareness will shorten the "bad" interconnects in terms of retiming potential and thus produce a solution similar to Placement 2 in Figure 1. In the iterative approach, the traditional timing-driven placer will shorten the critical path in the pre-retimed circuit and thus produce a solution similar to Placement 1 in Figure 1. In the iterative approach, another iteration of placement can be performed in the retimed circuit (Figure 1(c)) to further shorten path p_r. However, it is not as efficient as the simultaneous approach which can lead to the best solution in one round.

Clearly, the simultaneous approach has the advantage of efficiently generating better results. However, the existing simultaneous approaches have their limitations. In [2], retiming is integrated into the floorplanning stage based on the theory that as long as the target retiming preserves the number of FFs for every loop in a circuit whose underlying graph is strongly connected, there exists a valid finite sequence of retiming operations which can reach this target, thus making the loops become critical in terms of delay minimization when pipelining is to be performed afterwards. Edges in the critical loops are given high weights during the partitioning-based floorplanning stage to promote clusters of loops within a single partition, such that the clock period can be minimized (the clock period is determined by the ratio of the loop delay to the register count in the loop). However, in addition to that the complexity of such method is too high, there may not be enough global interconnects seen by a floorplanner, which could limit the retiming potentials. In [20] retiming is integrated into a simulated annealing (SA)-based placement for FPGA designs. At each temperature, critical cycle and slack analysis is performed so that the SA-based placement engine can be aware of the retiming possibility and thus reduce the lengths of the critical cycles. Their placement method, however, is based on the flat netlist and may have difficulty in handling large-scale designs. In [5], *sequential timing analysis* (Seq-TA), formerly called RTA, is proposed and integrated with a multilevel partitioning-based physical planner GEO. By doing that, GEO can generate results with retiming awareness. However, the proposed Seq-TA has one limitation — it can not handle gates (or clusters of gates) with multiple outputs, which makes it difficult to be directly applied to the multilevel framework. As a result, in GEO Seq-TA is always performed on the original single-output gate-level circuits, greatly affecting its efficiency.

In this paper, we present a practical solution for simultaneous retiming and multilevel global placement for performance optimization. Our contributions are (i) we generalize the Seq-TA to handle the gates/clusters with multiple outputs; (ii) we integrate retiming into the multilevel placement framework in order to efficiently handle large-scale designs and provide two speed-up techniques for Seq-TA to be efficiently integrated with the placement process.

The remainder of this paper is organized as follows. Section 2 reviews the related work and defines the terminologies. Section 3 describes the generalized sequential timing analysis for gates/clusters with multiple outputs. Section 4 describes the overall flow of integration. The experimental results are shown in Section 5, followed by the conclusions and ongoing work in Section 6.

2. REVIEW OF RELATED WORK

2.1 Retiming

Retiming is a sequential optimization technique that relocates the sequential elements in a circuit without changing the behavioral of the circuit. It moves the FFs across the combinational elements to optimize the clock period, the number of FFs, or power. In [11] Leiserson and Saxe first proposed a graph-theoretic model for a synchronous circuit. In this model, a circuit consisting of functional elements and globally clocked registers is transfered to a finite vertex-weighted, edge-weighted directed graph $G =< V, E, d, w >$. This graph is called a *retiming graph*. Vertex $v \in V$ corresponds to a functional element in the circuit and is weighted with its propagation delay $d(v)$. Edge $e \in E$ represents the interconnection between an output of a functional element and an input of another functional element, and is weighed with register count along the connection $w(e)$. Retiming of a circuit $G =< V, E, d, w >$ is an integer-valued vertex-labeling: $r : V \rightarrow Z$ such that G is transformed to a new graph $G_r =< V, E, d, w_r >$, where the edge weight w_r is defined for an edge $u \overset{e}{\rightarrow} v$ by $w_r(e) = w(e) + r(v) - r(u)$. Retiming a node by a value of i is an operation that removes i FFs from each fan-out edge and adds i FFs to each fan-in edge of that node. A circuit is retimed to a clock period ϕ by a retiming r if (i) $w_r(e) \geq 0$, (ii) $w_r(p) \geq 1$ for each path p such that its delay is greater than ϕ, where $w_r(p) = \sum_{e \in p} w_r(e)$. The retiming problem for clock period minimization can be formulated into a Mixed-Integer Linear Programming (MILP) problem and solved in polynomial time [11].

2.2 c-Retiming

Pan proposed *c-retiming* [15], a continuous version of retiming where the value assigned to a vertex can be a real number. To compute a *c-retiming* for a target clock period ϕ, another edge weight $l(e)$ for an edge $u \overset{e}{\rightarrow} v$ is defined as $-w(e)\phi + d(v)$, where $d(v)$ is the combinational delay of vertex v. The l-value of a node is defined as the weight of the longest path from the PIs to this node using the new edge weighting method. In a sequential circuit, if there is a PO whose l-value is greater than ϕ, the circuit can not be retimed to a clock period of ϕ. If, on the other hand, the l-values of all the POs are not greater than ϕ, the circuit can be retimed to a clock period less than $\phi + K$, where K is the maximum gate delay in the circuit. Because c-retiming can be computed much more efficiently than retiming and can be converted to a retiming by a simple rounding, it is combined with other optimization and synthesis techniques, such as FPGA mapping [17, 7], performance-driven clustering [16, 4], and partitioning [6], for a tight integration. Moreover, c-retiming is used as the basis for sequential timing analysis in the next subsection.

2.3 Sequential Timing Analysis (Seq-TA)

In [5], the concept of *sequential timing analysis* (formerly called RTA) was proposed. Similar to the static timing analysis, where arrival time, required arrival time, and slack are computed such that the critical paths are identified, sequential arrival time (SAT), sequential required time (SRT), and sequential slack are computed based on the concept of c-retiming. SAT of vertex $v \in V$ in terms of fan-in vertices is defined as

$$l(v) = \max\{l(u) - \phi \cdot w(e) + d(e) + d(v) |\; u \xrightarrow{e} v, e \in E\}$$

Similarly, SRT of v in terms of fan-out vertices is defined as

$$q(v) = \min\{q(u) + \phi \cdot w(e) - d(e) - d(v) |\; v \xrightarrow{e} u, e \in E\}$$

where $d(e)$ is the interconnect delay of edge e. The slack of v, denoted as $s(v)$, is defined as $q(v) - l(v)$. If the SATs for all POs are not greater than ϕ, then the target delay ϕ is called feasible. A Bellman-Ford variant shortest path algorithm RTA is proposed to determine whether the target clock period ϕ is feasible. A binary search is performed in order to find the minimum feasible clock period. If ϕ is feasible, based on the SAT and SRT value of the vertices, the ϵ-network for the given sequential circuit is derived and used for net weighting during the partitioning phase in GEO.

Obviously, Seq-TA is very helpful for the placers (floorplanners) because it can identify the critical path with retiming potentials and let them be aware of it. On one hand, by minimizing those critical paths, the placer (floorplanner) can achieve further delay reduction with the anticipation of retiming. On the other hand, retiming is not required to be performed during the placement (floorplanning), saving a great deal of runtime.

3. SEQUENTIAL TIMING ANALYSIS FOR COMPLEX NETWORKS

The first contribution of this work is to extend the sequential timing analysis for circuits consisting of multi-output gates.

DEFINITION 1. *An SO-gate is a combinational gate that has a single output and a uniform propagation delay.*

DEFINITION 2. *An MO-gate is a combinational gate that has multiple outputs and/or non-uniform propagation delays.*

DEFINITION 3. *A simple network is a circuit network that consists of only SO-gates.*

DEFINITION 4. *A complex network is a circuit network that consists of MO-gates and SO-gates.*

In reality, the MO-gates correspond to multi-output gates, such as adders and MUX. In the multilevel optimization scenario, clusters of SO-gates can also be regarded as MO-gates. Previous work in Seq-TA [5] only works on the simple network.

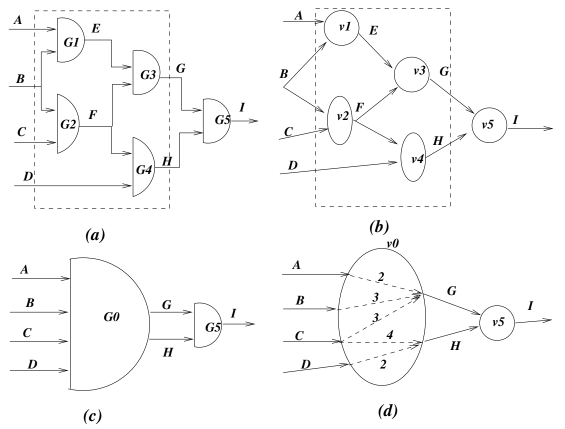

Figure 2: SO-gate vs. MO-gate. (a) is an example of a simple network and (b) is its corresponding retiming graph. (c) is an example of a complex network by clustering SO-gates into MO-gates and (d) is its retiming graph.

An example of SO-gates vs. MO-gates is shown in Figure 2. In Figure 2(a), all the gates have a single output and gate $G1$, $G3$ and $G5$ have a propagation delay of 1, while gate $G2$ and $G4$ have a propagation delay of 2. Its corresponding retiming graph is shown in Figure 2(b). When we cluster gate $G1$, $G3$, $G2$, and $G4$ into a cluster $G0$ (which can be regarded as an MO-gate), shown in Figure 2(c), $G0$ has not only multiple outputs but also non-uniform input-output propagation delays. Its corresponding retiming graph is shown in Figure 2(d). Obviously, it is necessary to extend the Seq-TA to the complex network, such that Seq-TA can be integrated with more optimization processes.

3.1 Generalized c-Retiming

When the MO-gates are combinational logic, i.e., there is no sequential logic (FFs) inside, we can generalize the c-retiming for them. In the retiming graph of a complex network, each gate (with either a single output or multiple outputs) corresponds to a vertex with internal edges from its inputs to its outputs according to the logic dependence. Each PI corresponds to a vertex with zero propagation delay, one input, and one output. Each PO corresponds to a vertex with zero propagation delay, one input, and one output. An example is shown in Figure 3. Vertex u corresponds to an MO-gate which has inputs $u_1^i, u_2^i, \ldots, u_n^i$ and outputs $u_1^o, u_2^o, \ldots, u_m^o$. $FO(u_n^i)$ is defined as the set of outputs of vertex u which are logically dependent on the input u_n^i of vertex u. We call output u_m^o a *reachable-output* of input u_n^i if $u_m^o \in FO(u_n^i)$. Similarly, $FI(u_m^o)$ is defined as the set of inputs of vertex u which can determine the logic value of output u_m^o of vertex u. u_n^i is called a *drive-input* of u_m^o if $u_n^i \in FI(u_m^o)$. The logic dependence is shown by the internal edges (the dashed edges in Figure 3) of u. An internal edge from an input u_s^i to its reachable-output u_t^o is denoted as $f^u_{I_s \to O_t}$ and its delay is denoted as $d(f^u_{I_s \to O_t})$. An edge e of the retiming graph (called external edge and shown by a solid line in Figure 3) connects an output of a vertex and an input of another vertex and its delay is denoted as $d(e)$.

DEFINITION 5. *l_1-value labeling: l_1-value of a vertex v is the weight of the longest path from PIs to this vertex using w_1 weight: $w_1(e) = -\phi \cdot w(e) + d(e) + d_{max}(v)$, where $d_{max}(v)$ is the maximum internal edge delays of vertex v. e is an external edge connecting with v. Each vertex has one l_1-value label.*

DEFINITION 6. *l_2-value labeling: l_2-value of an output v_t^o of*

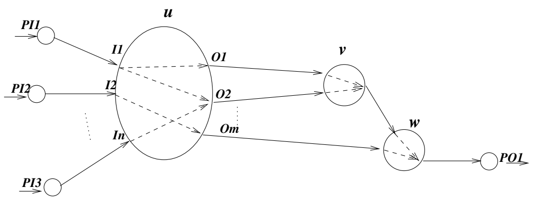

Figure 3: Retiming graph of a complex network.

vertex v is the weight of the longest path from all the PIs to this output using w_2 weight: $w_2(e, f^v_{I_s \to O_t}) = -\phi \cdot w(e) + d(f^v_{I_s \to O_t}) + d(e)$, where e is an external edge connecting with input v^i_s, $f^v_{I_s \to O_t}$ is an internal edge of v which e connects with via input v^i_s. Each output of a vertex has one l_2-value label.

COROLLARY 1. *For any vertex $v \in V, l_2(v^o_j) \le l_1(v)$, where v^o_j is an output of v.*

THEOREM 1. *In a sequential circuit with MO-gates, if $\exists$ a PO $v_o, l_2(v_o) > \phi$, then the circuit can not be retimed to a clock period of ϕ.*

THEOREM 2. *If $\forall$ PO $v, l_1(v) \le \phi$, then the circuit can be retimed to a clock period less than $\phi + K$, where K is the maximum input-output delay of all the gates/clusters.*

If ϕ_l is the minimum clock period such that the l_2-value labels of all the POs are not greater than ϕ_l, then ϕ_l is the lower bound of the feasible clock period achieved by retiming. If ϕ_u is the minimum clock period such that the l_1-value labels of all the POs are not greater than ϕ_u, then $\phi_u + K$ is the upper bound of the minimum feasible clock period achieved by retiming, where K is the maximum input-output delay of all the gates/clusters. For the circuits without MO-gates, l_1-value labeling and l_2-value labeling are equivalent and l_1-value labeling is equivalent to the SAT defined in Section 2.3.

Under the scenario of multilevel placement, the netlist of the original circuit is clustered to form a coarser netlist. Let C be a circuit which only consists of SO-gates. Let C_c be the circuit which is derived from C by clustering SO-gates into MO-gates.

THEOREM 3. *If the minimum clock period which can be achieved by retiming on circuit C is ϕ, and the minimum clock period which can be achieved by retiming on circuit C_c is ϕ_c, then $\phi \le \phi_c$.*

Due to page limit, we have to omit the proofs for the above theories. Please refer [8] for details.

3.2 Generalized Sequential Timing Analysis

Based on the generalized c-retiming, we can generalize the sequential timing analysis for the complex network by using the l_2-value labeling to compute SAT and SRT values for the inputs and outputs of vertices. We define the SAT for each output v^o_i of vertex v (shown in Figure 4) in the retiming graph G as

$$\begin{aligned} SAT(v^o_i) = \quad & \max\{SAT(u^o_s) - \phi \cdot w(e) + d(e) + d(f^v_{I_t \to O_i}) \\ & \mid v^i_t \in FI(v^o_i), u^o_s \xrightarrow{e} v^i_t, e \in E\} \end{aligned}$$

Similarly, SRT of each input u^i_j of vertex u is defined as

$$\begin{aligned} SRT(u^i_j) = \quad & \min\{SRT(v^i_t) + \phi \cdot w(e) - d(e) - d(f^u_{I_j \to O_s}) \\ & \mid u^o_s \in FO(u^i_j), u^o_s \xrightarrow{e} v^i_t, e \in E\} \end{aligned}$$

The edge slack is defined as $slack(e) = SRT(v^i_t) - d(e) - SAT(u^o_s)$ where $u^o_s \xrightarrow{e} v^i_t, e \in E$.

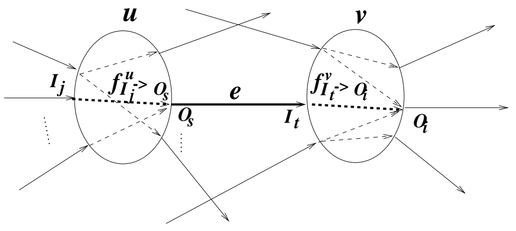

Figure 4: Illustration of generalized SAT and SRT computation

We call the generalized sequential timing analysis Seq-TA-M. If the SATs of all the PO inputs are not greater than ϕ, then the target clock period ϕ is called feasible under the l_2-value labeling. Though l_2-value c-retiming can not be converted to a retiming, it gives us enough information to catch the critical "sequential path" that should be minimized during the placement phase such that the placement solution can produce the best result achieved by retiming. We still use the Bellman-Ford variant shortest path algorithm to determine whether the target clock period ϕ is feasible under the l_2-value labeling and compute the generalized SAT and SRT for inputs and outputs of the vertices, as the RTA algorithm [5] does. We start with the initialization for the SAT and SRT values. SAT for all the PI outputs are set to zero while the outputs of other vertices are set to $-\infty$. SRT for all the PO inputs are set to ϕ while the inputs of other vertices are set to ∞. During one iteration of relaxation, we visit the vertices and iteratively update SAT and SRT values of their inputs and outputs. The iteration stops when SATs and SRTs converge to their maximum and minimum values, respectively, or there is one PO input whose SAT is greater than ϕ. A binary search is performed to find the minimum feasible clock period under the l_2-value labeling. Based on the Seq-TA-M, we can identify the ϵ-network which is defined to be a subcircuit consisting of external edges whose slacks are smaller than or equal to ϵ in the current placement. The ϵ-network consists of critical interconnects, in terms of the retiming potential, that deserve attention from the placement engine for optimization.

4. INTEGRATION OF RETIMING WITH MULTILEVEL PLACEMENT FRAMEWORK

The second contribution of our work is to provide a solution for integrating retiming to a multilevel placement framework and two speed-up techniques for Seq-TA-M.

The multilevel optimization method is very powerful in solving problems with high computation complexity. It includes two phases, a coarsening phase and a refinement phase. We integrate retiming with a multilevel coarse placement [1] by performing Seq-TA-M to identify critical nets and assigning higher weights to them. A simulated annealing-based placement engine minimizes the weighted wirelength to reduce the length of the critical path and thus to reduce the delay. The integrated placement algorithm, called mPG-rt, consists of a bottom-up coarsening phase and a top-down placement. The overall flow is shown in Figure 5.

During the bottom-up coarsening phase, we build a coarser level netlist(graph) L^i from level L^{i-1} by performing clustering until we reach level L^t where the number of clusters is within a certain range, so that the SA-based placement can be efficiently performed. We use the *FirstChoice* (FC) clustering algorithm [9] as it experimentally generates a better hierarchy for global placement [1]. FFs are not allowed to be clustered if Seq-TA-M is to be performed at

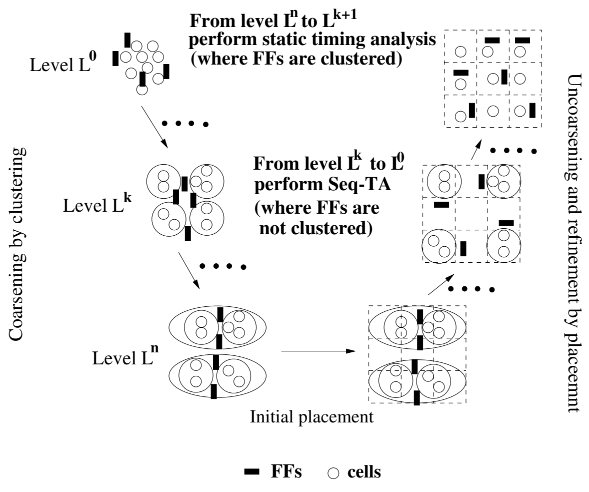
From level L^n to L^{k+1} perform static timing analysis (where FFs are clustered)
Level L^0
From level L^k to L^0 perform Seq-TA (where FFs are not clustered)
Level L^k
Coarsening by clustering
Uncoarsening and refinement by placemnt
Level L^n
Initial placement
FFs
cells

Figure 5: Overall flow of mPG-rt

a certain level, because Seq-TA-M can not handle clusters with internal FFs. We allow FFs to be clustered above level L^k, where k is a user-defined parameter, for the following reasons: (i) at coarser levels, the sizes of clusters are fairly large compared to those at the finer levels. Moving FFs across such huge clusters may not bring delay reduction, as retiming is an optimization for a fine-granularity structure; (ii) forbidding FFs to be clustered at all levels may bring too much constraint in the coarsening phase and thus may cause bad clustering results and a poor hierarchy for refinement.

During the top-down placement refinement phase, at level L^i, where $i > k$, i.e., FFs may be clustered, we perform static timing analysis and assign weight to nets according to their criticality to reduce the longest path delay at the current level. This is helpful for reducing the runtime for performing Seq-TA-M at the finer levels as it brings Seq-TA-M a placement with a decent path delay to start with during the binary search for finding the minimum feasible clock period. At level L^i, where $i \leq k$, i.e., FFs are not clustered, we build a retiming graph and perform Seq-TA-M once at each temperature, and weight each net according to its criticality in terms of the retiming potential. Except for the top level, where a full-scale SA process is performed, the SA process starts from a low temperature at the following levels to save the runtime [1]. At the finest level, after the SA-based placement is finished, final retiming is performed and FFs are inserted as proposed in [5]. A low temperature SA process may be required to legalize the retimed placement.

4.1 Speed up the Sequential Timing Analysis

Although the Seq-TA-M is polynomial, the complexity can still be up to $O(|V||E|)$ when determining whether the target delay is feasible under the l_2-labeling and computing SAT and SRT. Additionally, the binary search for finding the minimum feasible clock period under the l_2-labeling may also be time-consuming. When we perform Seq-TA-M, we visit the vertices in their pseudo-topological order, as it is shown in [15], that if we visit the vertices in the pseudo-topological order, the relaxation will be converged much faster than one with a random order. Furthermore, we provide two methods to further speed up the Seq-TA-M process. The first method is called "single-ϕ," that is, instead of doing a binary search to find the minimum feasible clock period for SAT, SRT and slack computation, we just use the longest path delay D_{max} of the current placement to calculate the SAT, SRT, and slack. Because D_{max} is the longest path delay of the given placement, it is feasible under l_2-value labeling. The second method is called "early abortion," that is, for each target clock period ϕ_i, we only perform k iterations to determine whether ϕ_i is feasible. If the SAT computation can not converge within k iterations, ϕ_i is regarded as infeasible, though it may actually be feasible if more iterations are allowed. By doing that, we can reduce the runtime, though we may get a less accurate feasible clock period estimation.

circuit	#GA	#PI	#PO	#FF
s9234	1290	28	39	135
s5378	1443	35	49	163
s13207	3146	59	152	486
s15850	3784	76	150	515
s38584	13209	38	304	1423
ind1	29780	2630	3242	603
ind2	26060	2772	6242	1755
ind3	52197	2801	3070	2001
ind4	101531	4155	4547	8333

Table 1: Benchmark circuit characteristics.

4.2 Net Weighting

At each temperature in the SA process, once either static timing analysis or sequential timing analysis is performed, the criticality of nets is obtained and transfered to weights on the nets. When static timing analysis is performed, we use the PATH algorithm [10] to compute the weights for nets. When the sequential timing analysis is performed, we use the net weighting methods proposed in [12, 19] to compute the weight. The delay cost of an edge in the timing graph (or retiming graph) is the product of the edge delay and its weight. Delay_Cost is the sum of all the delay costs of all the edges/connections in the timing graph (or retiming graph). Wire_Cost is the sum of all the bounding box lengths of the nets. The overall cost is a weighted sum of Wire_Cost and Delay_Cost defined as

$$Cost = \alpha \frac{Delay\_Cost^{current}}{Delay\_Cost^{previous}} + (1-\alpha)\frac{Wire\_Cost^{current}}{Wire\_Cost^{previous}}$$

α is a user defined value which can trade off wirelength with delay. We set it to 0.5.

5. EXPERIMENTAL RESULTS

We implemented our algorithm mPG-rt in C++/STL and tested it on a Sun Blade 1000 workstation running at 750MHz frequency. The benchmark consists of 5 ISCAS circuits and 4 large scale industrial designs which were used in [5]. [1] The delay model we used is the same as that in [5]. [2] The circuit characteristics are listed in Table 1.

5.1 Impact of Speedup Techniques of Seq-TA

We tested our two speed-up techniques, "single ϕ" and "early abortion," for Seq-TA-M on some of the largest circuits on a 8x8 global placement grid. The results are shown in Table 2. For each circuit we adopted "single ϕ" with full relaxation, "early abortion" with iteration of 5, 15, and 30 and the full relaxation.

From this table it can be seen that using a limited number of iterations can greatly reduce the runtime with a reasonable quality

[1]Currently the large-scale placement benchmarks in public domain are mainly for wirelength-driven placement and are lack of functionality information of the cells/blocks which is required by the retiming operation.

[2]We did not compare mPG-rt with GEO because GEO did not consider wirelength minimization and a direct comparison in terms of wirelength may not be very meaningful.

circuit	single ϕ		binary search min. ϕ							
	full relaxation		early abortion						full relaxation	
			5 iter.		15 iter.		30 iter.			
	dly	cpu	dly	cpu	dly	cpu	dly	cpu	dly	cpu
s38584	36	713	36	803	32	837	32	911	32	1228
ind1	353	3529	353	3649	353	3916	353	4337	353	4793
ind2	59	3187	59	3367	54	3654	46	3927	46	5343
ind3	550	5858	550	5903	550	6378	550	6886	546	14392
ind4	102	15818	102	16714	106	20265	102	21474	95	84316
avg.	1.10	0.50	1.10	0.53	1.06	0.57	1.02	0.62	1	1

Table 2: Impact of speedup techniques for Seq-TA-M

circuit	GP	delay			wirelength		runtime(s)	
		mPG		mPG-rt	mPG	mPG-rt	mPG	mPG-rt
	grid size	before retiming	after retiming		+ retiming		+ retiming	
s9234	8x8	34	25	23	415	449	79	163
s5378	8x8	24	17	15	486	528	97	195
s13207	8x8	43	39	33	1422	1512	289	428
s15850	8x8	63	48	39	1437	1551	314	574
s38584	8x8	44	43	36	7063	7177	2387	3411
ind1	16x16	348	330	326	9719	10309	4148	9216
ind2	16x16	51	40	40	9947	10056	4158	7200
ind3	16x16	588	536	500	25151	26936	7801	14113
ind4	16x16	104	101	88	17826	25246	25040	38313
avg.		1	0.86	0.77	1	1.10	1	1.79

Table 3: Impact of simultaneous retiming and placement

loss. It becomes even more efficient when the circuit size increases. Therefore in all of our experiments shown in the next subsection we used the "early abortion" scheme with iteration of 30.

5.2 Impact of Simultaneous Retiming and Placement

Our multilevel coarse placement can be used as a wirelength-driven placer when the cost is totally set to be the Wire_Cost (which is mPG). We ran mPG [1] followed by retiming and a post legalization on the global placement. [3] We also ran mPG-rt followed by the same post legalization procedure. We compared the results generated by mPG followed by retiming with those generated by mPG-rt to show the impact of simultaneous retiming and placement. We also report the delay of placement results generated by mPG before retiming to show the impact of retiming on placement. The comparison results are shown in Table 3.

It can be seen that (i) retiming can improve the performance by 14% on average when it is applied after placement; (ii) our simultaneous approach for retiming and placement can outperform the two-step approach (placement followed by retiming) by 10% on average in terms of delay with 10% wirelength increase, demonstrating the necessity of such integration.

6. CONCLUSIONS AND ONGOING WORK

We proposed a practical solution for integrating retiming into the multilevel global placement for large-scale designs. We extended the available sequential timing analysis to handle gates/clusters with multiple outputs, and integrated it into a multilevel SA-based placement framework for performance optimization. We also provided two speed-up techniques to enable it to be efficiently integrated into the placement engine. Experimental results show that such an approach is efficient compared with the two-step approach (placement followed by retiming). We are currently working on large benchmarks to further test our approach, and plan to use a performance-driven cluster algorithm in the coarsening phase instead of using connectivity-driven clustering algorithms.

[3]It is done by a timing-driven SA-based refinement on the finest level in the mPG framework.

Acknowledgments

This work is supported in part by Semiconductor Research Corporation under Contract 2001-TJ-910 and a grant from Fujitsu Laboratories of America under the California MICRO program.

7. REFERENCES

[1] C.-C. Chang, J. Cong, D. Pan, and X. Yuan. Physical hierarchy generation with routing congestion control. In *Proc. Int. Symp. on Physical Design*, pages 36–41, 2002.

[2] P. Chong and R. K. Brayton. Characterization of feasible retimings. In *Proc. Int. Workshop on Logic and Synthesis*, pages 1–6, 2001.

[3] J. Cong. An interconnect-centric design flow for nanometer technologies. In *Proceedings of the IEEE*, pages 505–527, April 2001.

[4] J. Cong, H.-C. Li, and C. Wu. Simultaneous circuit partitioning/clustering with retiming for performance optimization. In *Proc. Design Automation Conf*, pages 460–465, 1999.

[5] J. Cong and S. K. Lim. Physical planning with retiming. In *Proc. Int. Conf. on Computer Aided Design*, pages 2–7, 2000.

[6] J. Cong, S. K. Lim, and C. Wu. Performance driven multi-level and multiway partitioning with retiming. In *Proc. Design Automation Conf*, pages 274–279, 2000.

[7] J. Cong and C. Wu. An efficient algorithm for performance-optimal FPGA technology mapping with retiming. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 17(9):738–748, 1998.

[8] J. Cong and X. Yuan. Multilevel global placement with retiming. Technical Report 03-0023, Computer Science Department, University of California, Los Angeles, 2003.

[9] G. Karypis and V. Kumar. Multilevel k-way hypergraph partitioning. In *Proc. Design Automation Conf*, pages 343–348, 1998.

[10] T. Kong. A novel net weighting algorithm for timing-driven placement. In *Proc. Int. Conf. on Computer Aided Design*, pages 172–176, 2002.

[11] C. E. Leiserson and J. B. Saxe. Retiming synchronous circuitry. *Algorithmica*, pages 5–35, 1991.

[12] A. Marquardt, V. Betz, and J. Rose. Timing-driven placement for FPGAs. In *Proc. ACM/SIGDA Int. Symp. on Field Programmable Gate Arrays*, pages 203–213, 2000.

[13] I. Neumann and W. Kunz. Placement driven retiming with a coupled edge timing model. In *Proc. Int. Conf. on Computer Aided Design*, pages 95–102, 2001.

[14] I. Neumann and W. Kunz. Tight coupling of timing-driven placement and retiming. In *Proc. IEEE Int. Symp. on Circuits and Systems*, pages 351–354, 2001.

[15] P. Pan. Continuous retiming: Algorithms and application. In *Proc. IEEE Int. Conf. on Computer Design*, pages 116–121, 1997.

[16] P. Pan, A. K. Karandikar, and C. L. Liu. Optimal clock period clustering for sequential circuits with retiming. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 17(6):489–498, 1998.

[17] P. Pan and C.-C. Lin. A new retiming-based technology mapping algorithm for LUT-based FPGAs. In *Proc. ACM/SIGDA Int. Symp. on Field Programmable Gate Arrays*, pages 35–42, 1998.

[18] Semiconductor Industry Association. *International Technology Roadmap for Semiconductors*, 2002 update.

[19] M. Senn, U. Seidl, and F. Johannes. High quality deterministic timing driven FPGA placement. In *Proc. ACM/SIGDA Int. Symp. on Field Programmable Gate Arrays*, 2002.

[20] D. P. Singh and S. D. Brown. Integrated retiming and placement for field programmable gate arrays. In *Proc. ACM/SIGDA Int. Symp. on Field Programmable Gate Arrays*, pages 67–76, 2002.

[21] T. C. Tien, H. P. Su, and Y. W. Tsay. Integrating logic retiming and register placement. In *Proc. Int. Conf. on Computer Aided Design*, pages 136–139, 1998.