Performance-driven Global Placement via Adaptive Network Characterization

Mongkol Ekpanyapong and Sung Kyu Lim

School of Electrical and Computer Engineering,
Georgia Institute of Technology
{pop,limsk}@ece.gatech.edu

Abstract

Delay minimization continues to be an important objective in the design of high-performance computing system. In this paper, we present an effective methodology to guide the delay optimization process of the mincut-based global placement via adaptive sequential network characterization. The contribution of this work is the development of a fully automated approach to determine critical parameters related to performance-driven multi-level partitioning-based global placement with retiming. We validate our approach by incorporating this adaptive method into a state-of-the-art global placer GEO. Our A-GEO, the adaptive version of GEO, achieves 67% maximum and 22% average delay improvement over GEO.

1. Introduction

Placement problem can be classified into two classes: global placement and detailed placement. Global placement identifies the location where groups of cells should be located. Whereas detailed placement provide detailed location for each cells such that the global placement solution is preserved. Recently global placement plays a significant role due to the increasing in circuit constraints and complexities. There are three major approaches in global placement: mincut-based approaches [4,13,27,2,5], analytical approaches [10,15], and Simulated Annealing approaches [24,25]. Mincut-based uses top down approach to recursively partition circuits into sub-netlists and assign gates to the tiles. Based on its fast running time and flexibility in handling various constraints, it has been adopted in many modern state-of-the-art placements, including a state-of-the-art timing driven placement [8].

With tremendous increasing demand in high performance computing, circuit performance improvement during physical design becomes highly interesting. During physical planning, gate location can be identified and hence can be used to accurately calculate wire delay. Knowing both gate and wire delay, total delay for the entire circuit can be computed. Circuit optimization on this physical design level can employ this knowledge and gain superior performance over same optimizations without such information. One optimization that can employ this advantage is retiming [17].

Retiming is a logic optimization technique, which shifts the position of flip-flops (FFs) for delay minimization or FFs reduction [17]. Recently, retiming has become more attractive in physical design where wire delay is more essential in the context of deeper submicron technology [23,12]. Exploiting geometric information enables us to further enhance retiming techniques with floorplanning; since location information is available, thus allows more accurate wire delay calculation. Retiming in physical design can be classified into two approaches: iterative approach and simultaneous approach. The iterative approach [26,18,19] first performs placement or floorplanning, after that retiming is performed. The alternative approach [8,6,22,9] simultaneously performs placement or floorplanning with retiming, by incorporating retiming information during placement or floorplanning. In [9], the authors suggest that the latter approach is better than the first with respect to retiming delay improvement.

In [8], a state-of-the-art approach for mincut-based placement with retiming, so called GEO, was proposed. The concepts of Sequential Arrival Time (SAT) [21] and Sequential Required Time (SRT) were adopted here. Then slack value, used to identify critical gates/clusters, can be computed as the difference between SRT and SAT. Subsequently, an ε-network which contains the set of critical cells can be identified. By assigning additional delay weight α to an ε-network, those critical cells tend to be grouped closer together during circuit partitioning. Cong et al., [9], extend [8] work by generalizing the model to handle the gates/clusters with multiple outputs. However, both approaches keep best weighted-cutsize among all runs, which is the cutsize that incorporates retiming information as the best result to the next floorplanning level. In this paper, we show that while weighted-cutsize highly correlates with retiming delay, there is no guarantee that it will result in best retiming delay among all runs. Next we propose a methodology to identify the weight α assigned to those critical cells/nets instead of using a fixed constant value such as in [8]. Furthermore, we suggest a way to properly identify critical cells such as ε parameter in [8,9], sometimes referred to as K paths in [1], or defined to be at least 90% of the critical path delay as in [3]. In [22], the authors propose a way to identify critical edges using criticality distribution. However with their method, one has to iteratively search until finding the factor value making 90-100% critical to no more than 5% of the total number of edges. This is time

consuming and for some circuits, it is hard to achieve. In [16], the author suggests another way to compute net weight called PATH, however PATH requires computing exponential function, which requires more computational time. Hence, a further contribution of this paper is a simple method to compute net weight by simply counting critical cells inside each net. Finally, we study the impact of clustering. When we perform the physical design on big circuits, clustering [14,7] becomes necessary because of its fast partitioning and its success in cutsize reduction. However employing clustering, we lose accuracy of circuit retiming information since it is calculated based on each gate location. To alleviate this, we propose a way to adaptively decide when to perform clustering based on circuit information.

The organization of this paper is as follows. Section 2 describes problem formulation. Our observations are discussed in section 3. Section 4 is devoted to our methodology. Section 5 presents our experimental results and final section presents our conclusion and future work.

2. Problem Formulation

Given a sequential gate-level netlist $NL(C, N)$, where C is the set of cells representing gates and flip-flops, and N is the set of nets connecting the cells, the purpose of the Physical Planning with Retiming (PPR) problem is to assign cells in NL to a given m x n (=K) slots by preserving area constraints. Given a PPR solution $C \rightarrow B$, let $\omega(B)$ and $\phi(B)$ respectively denote the wirelength and retiming delay (all of them to be defined later). The formal definitions of PPR problem is as follows:

PPR Problem The Physical Planning with Retiming problem has a solution P: $C \rightarrow B$, when each cells in C is assigned to a unique block. $B = \{B_1(x_1,y_1), B_2(x_2,y_2),..., B_K(x_K,y_K)\}$, where B denotes the set of blocks, and (x_i,y_i) represents the geometric locations of B_i, and area constraints $A(L,U)$, for $1 \le i \le K$. PPR solution has to satisfy the following condition: 1) $B_i \subset C$ and $L \le |B_i| \le U$ 2) $B_1 \cup B_2 \cup \ldots \cup B_k = C$ 3) $B_i \cap B_j = \emptyset$. The primary objective of PPR is to minimize $\phi(B)$ and secondary objective is to minimize $\omega(B)$.

2.1. Delay Objective

For the delay objective, we model NL using a directed graph $G = (V, E)$ where the vertex set V represents cells, and the directed edge set E represents the signal direction in NL. In the *geometric delay model*, each vertex v has delay $d(v)$ and each edge $e=(u,v)$ has delay $d(e)$. Let $s(e)$ denote the *cut-state* of e: $s(e)=1$ if e is cut, and $s(e)=0$ otherwise. In this paper, we assume $d(e) = m(e) \cdot s(e)$, where $m(e) = |x_u - x_v| + |y_u - y_v|$. The delay of a path p, denoted $d(p)$, is the sum of the delay of gates and edges along p. Then, the *normal delay* $\delta(B)$ of global placement solution B is computed as $\max_{p \in G}\{d(p(u,v)) \mid u \in PI \text{ or } FF \ \& \ v \in PO \text{ or } FF\}$.

By employing the concept of a retiming graph [17], we model NL using a directed graph $R = (V, E_R)$, where the edge weight $w(e)$ of $e=(u,v)$ denotes the number of flip-flops between gate u and v. The path weight can be calculated by $w(p)=\sum_{e \in p} w(e)$. Let $w^r(e)$ denote edge weight after retiming r, i.e. number of flip-flops on the edge after retiming. Then, $w^r(p)=\sum_{e \in p} w^r(e)$. A circuit is retimed to a delay ϕ by a retiming r if the following conditions are satisfies; (i) $w^r(e) \ge 0$ for each e, (ii) $w^r(p) \ge 1$ for each path p such that $d(p) > \phi$. We define the edge length of $e=(u,v)$ as $l(e) = -\phi \cdot w(e) + d(v) + d(e)$, and the path length of p as $l(p)= \sum_{e \in p} l(e)$. The *sequential arrival time* of vertex v, denoted $l(v)$, is the maximum path length from PIs or FFs to v. If the sequential arrival time of all POs or FFs are less than or equal to ϕ, the target delay ϕ is called *feasible*. Let $q(e) = \phi \cdot w(e) - d(u) - d(e)$, is the required edge length of e. The required path length $q(p)= \sum_{e \in p} q(e)$. The *sequential required time* of vertex v, denote $q(v)$ is the minimum required path length from v to POs or FFs, when $q(\text{PO})$ or $q(\text{FF}) = \phi$. Then slack of v is given by $q(v)$-$l(v)$. Let $D_g = \max\{d(v) \mid v \in V\}$. Then, the *retiming delay* $\phi(B)$ of a partitioning and/or floorplanning solution B is the minimum feasible $\phi + D_g$.

2.2. Wirelength Objective

We model net-list NL using a hypergraph $H=(V, E_H)$, where the vertex set V represents cells, and the hyperedge set E_H represents nets in NL. Each hyperedge is a non-empty subset of V. The x-span of hyperedge h, denoted h_x, is defined as $h_x = \max_{c \in h}\{x_i \mid c \in B_i\} - \min_{c \in h}\{x_i \mid c \in B_i\}$. The y-span, denoted h_y, is calculated using the y-coordinates. The sum of x-span and y-span of each hyperedge h is the half-parameter of the bounding block (HPBB) of h and denoted $HPBB(h)$. The *wirelength* $\omega(B)$ of global placement solution B is the sum of HPBB of all hyperedges in H.

3. Observations

In this section we provide some observed information on retiming based on timing analysis (RTA) [17] using gate location based on circuit characteristics and mincut-based global placement. In addition, we can employ such information to dynamically guiding the mincut-based global placement for performance improvement. We perform the study on circuits from ISCAS89 [29] and ITC99[28] suites. We assume unit delay for all gates in the circuits. Table 3.1 shows the statistical information of benchmark circuits. We provide the number of gates, PIs, POs and FFs for each circuit. Dr represents retiming delay. Here it is the lower bound of retiming delay, which is calculated by assigning zero delay to all edges and then performing RTA. Ob represents the observed circuits that are used toward our

Table 3.1. Benchmark circuit characteristics.

ckt	gate	PI	PO	FF	Dr	Ob
s641	379	35	42	19	74	y
s820	289	18	24	5	10	y
s1196	529	14	32	18	24	y
s1238	508	14	32	18	22	y
s1494	647	8	25	6	16	y
s5378	2828	36	49	163	32	y
s9234	5597	36	39	211	39	y
s13207	8027	31	121	669	50	y
s15850	9786	14	87	597	62	y
s35932	16353	35	2048	1728	27	y
s38417	22397	28	106	1636	32	
s38584	19407	12	278	1452	47	
b14o	5401	32	299	245	27	
b15o	7092	37	519	449	38	
b17o	22854	37	97	1414	38	
b20o	11979	32	22	490	44	
b21o	12156	32	22	490	43	
b22o	17351	32	22	703	46	

observations, when 'y' indicates that it is used in our observations. In section 5, we perform experiments on all circuits to show that our study is not biased toward only the observed circuits. Throughout the paper, our studies are based on 8x8 slots with 5 runs, $\alpha = 20$, T filter, and ε = top 5% unless explicitly specified (all to be defined later).

3.1. Correlation between weighted cutsize and retiming

Most simultaneous placement or floorplanning with retiming [8,9] employs weighted cutsize during partitioning that is considering retiming based timing information during net weight computation. For example, in GEO [8], Net Weight = Cutsize Weight + α·Delay Weight is used. The α parameter determines how important delay weight is comparing with cutsize. Here we compute the correlation between weighted cutsize and retiming delay and we get

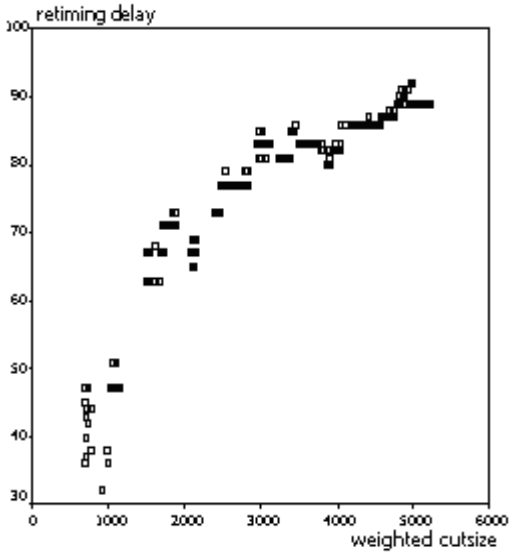

Figure 3.1 Correlations between retiming delay and weighted cutsize on s1238

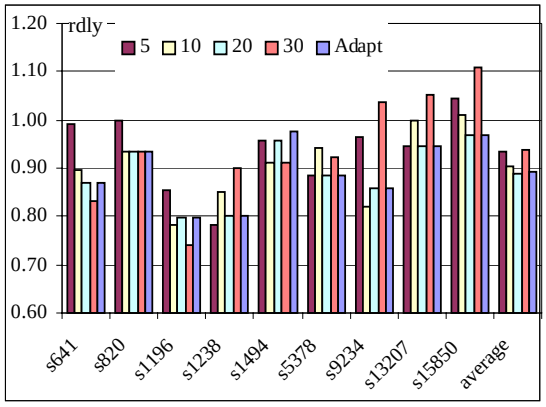

Figure 3.2 Impact on different α value

about 0.9 on average. This implies that weighted cutsize is highly related to retiming delay. Then we plot the retiming delay versus weighted cutsize. Results from the scatter plots of all observed circuits are similar. Figure 3.1 shows the plot on s1238 circuit on 8x8 global placement. From the plot, when the first bipartition has a weighted cutsize at about 800, the next partitions are about 1,200 and 1,600 and so on. If we look at the first partition, the retiming delay ranges from 36 to about 47, even though the weighted cutsizes are close together. From the result, it implies that using weighted cutsize alone might not be enough to achieve a high reduction in retiming delay even though the weighted cutsize is highly correlated with retiming delay.

3.2. Delay weight (α parameter)

As shown in Figure 3.1, the highest impact partition based on mincut-based approach is the first partition. The gap between each subsequent partition becomes closer when we go down into lower levels. For example in that figure, when cutsize is about 5,000 the retiming delay gaps are about 1 or 2 different. Hence we can assign higher α value and then reduce it when we go into more detailed partitions. Here we first find the best α as shown in Figure 3.2. The plot is using the best retiming for each run as discussed in section 3.1 and is normalized based on value $\alpha = 1$. We found that $\alpha = 20$ provides the best retiming delay on average. Next we show that the result using adaptive α is

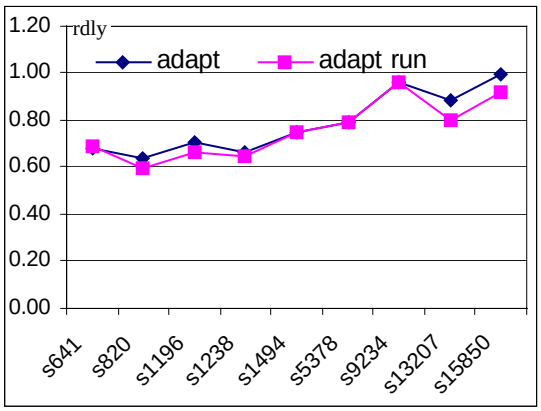

Figure 3.3 impact of adaptive number of run

close to a fixed α value and only one circuit has a higher value (comparing with α = 20). Whereas we can reduce computation time about 5% by setting α = 0 (i.e. consider only cutsize improvement when partitioned node is bottommost node of the root's right sub-tree). Otherwise we use α = 20. Note that we still project best retiming delay.

3.3 Number of Runs

Based on mincut-based global placement characteristic shown in Figure 3.1, if we pay more attention on the earlier partitioning, better result can be gained since there exists more room for retiming delay improvement. Here, we propose a way to adapt the number of runs. Instead of using a fix number of runs throughout the program, we will begin with greater number of runs in the earlier partitions and then reduce it subsequently. For example, consider 8x8 global placement, instead of using fixed 5 runs throughout the program, we can modify it to 20 runs in the first 3 levels and 3 runs for the rest. Hence we have 7x20 + 3x56 = 308 < 63 x 5 = 315. The results in Figure 3.3 shows that even though we have fewer number of runs, we can gain a better improvements almost all the benchmarks (s641 is the sole exception).

3.4. Nets filtering and Cells selection (ε parameter)

During circuit partitioning, hypergraph model is employed. In [8], the net weight will be assigned based on criticality among cells/clusters that net is covered. The equation used in [8] to assign net weight is as follows:

$$dwgt(n) = 1 - \frac{\min\{slack(v) \mid v \in n\}}{\max\{slack(w) \mid w \in NL\}}. \qquad \text{(Eqn1)}$$

However from the equation, non-critical nets might be included. Considering example from Figure 3.4, suppose cells *a-e* is in the critical path. By selecting nets that have critical cells inside, we will include other cells such as *f,g,h,i,j,* and *k* that are not critical. To remedy this problem, [9] employs PATH [16] weight function. However it is expensive since PATH requires the computation of exponential function. Here we proposed two net filtering methods. The first one is T filter. In T filter, the net is considered critical only when at least two cells/clusters are critical. This can eliminate cells *f,g,h,i, j* and *k* in Figure 3.4. Also we can still use the *dwgt* weight function as discussed earlier. The second filter is A filter, the net is included only when all cells/clusters in the net are critical. Figure 3.5 shows the impact of different filtering methods where O represents GEO, T represents T filter method, and A represents A filter method. The Graph is normalized relative

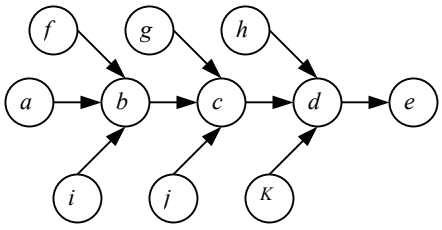

Figure 3.4 Example of net filtering

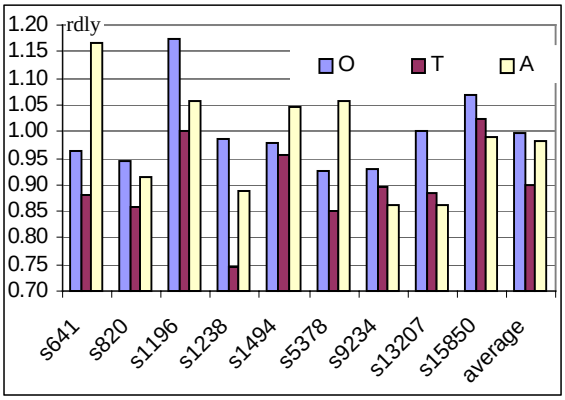

Figure 3.5 Impact on different net filtering

to GEO with no cut threshold (i.e. ε = 100%) whereas the rest use cut threshold ε = 5% of cells/clusters (i.e. top 5% minimum slack value as critical cells/clusters). Results show that on average, the T filter method is better than the GEO and the A filter by about 11% and 9% respectively. However, if we look at the circuits with a large number of cells such as s9234, s13207, and s15850, the A filter yields better result. This is because it is harder to group large critical cells/clusters into the same partition when the number of cells/clusters is large. The A filter then can be used to consider only very critical cells/clusters. Here we propose a threshold deciding when to use the A filter: when the number of cells in current partitioned circuits is higher than 5,000 cells (based on results from the graph).

Next we study a way to dynamically adapt an ε parameter based on circuit characteristic. The ε parameter is the parameter identifying how many cells considering being critical e.g. first top 5% cells having minimum slack value. In [8,1,3,9], it is assigned as a fixed value. We first classify slack cumulative normalized frequency distribution (normalized to 100%) into three groups as slow start, medium start, and fast start based on the number of cells/clusters falling into top minimum slack value as shown in Figure 3.6[1] respectively. The boundary lines we use here are 15% for medium and 35% for fast start based on results

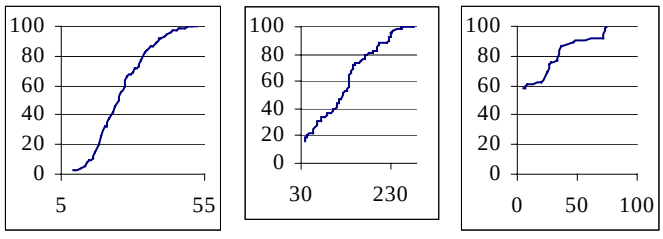

Figure 3.6 slack distribution classifications when x-axis represents slack value and y-axis represents cumulative frequency percentage normalized to 100%

[1] Here we plot slack distribution using initial clock cycle, hence minimum slack will not have zero value.

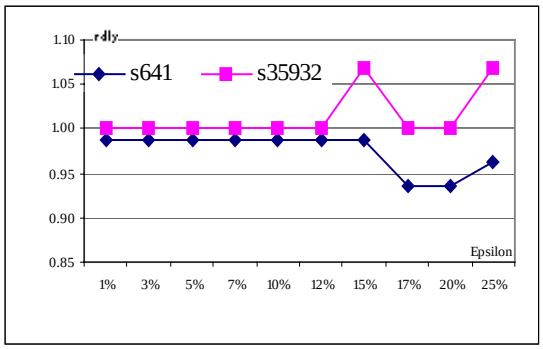

Figure 3.8 Examples of medium and fast start

from our observed circuits. Since there is randomness involved different initial partition can result in different slack distribution. From our observation, most circuit's initial partitions fall into the slow start category. We found that for slow start distribution, choosing the best ε value is difficult; especially where randomness is involved. The result in Figure 3.7 shows that the higher the number of runs is the lower the variance is on different ε values. In Figure 3.7, 5r and 20r represents 5 and 20 runs respectively, and the percentage represents the ε value. The graph is normalized relative to GEO with ε=100%. We found that the variance reduces from 0.0021 to 0.0008 when the number of runs increases from 5 to 20 runs. We found that when the slack distribution of current partition falls into medium start category, selecting the first critical slack value is sufficient since it already contains a substantial number of cells/clusters for consideration. For the fast start distributions, assigning α to be zero (i.e. consider only minimizing wire length) is adequate, since there are too many critical cells/clusters, and it is hard to group these cells/clusters into the same partition without violating area constraints. Figure 3.8 shows the impact on the medium and the fast start case on bipartition with 5 runs on 2x1 slots normalized to global placement that targeting cutsize only

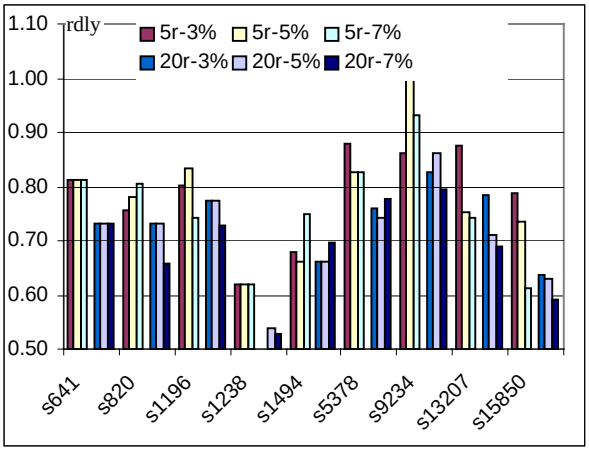

Figure 3.7 Impact on different ε value

i.e. setting α = 0. When s641 slack distribution is shown in Figure 3.6b and s35932 slack distribution is shown in Figure 3.6c. For the medium start, once ε is higher than starting threshold, the retiming delay drops as can be seen when ε = 17%. On the other hand, for the fast start case, no matter what value of ε is, retiming delay is higher than partitioning targeting cutsize.

Since the wire delay plays major role in deeper submicron technology [23, 12], given a circuit the distribution depends on both cell connectivity and wire delay. However wire delay has randomness involved since we adopt randomness during initial partition. Despite the fact that we can enforce an initial partitioning instead of a random partition, it will limit our search space and can have a larger impact on wirelength.

3.5 Clustering

Circuit clustering is important for cutsize reduction especially when considering a large number of cells. This also holds for weighted cutsize. As shown earlier in section 3.1, that weighted cutsize is correlated to retiming delay. Hence for large circuit without clustering, it will result in poorer performance. However by clustering the circuit, the accuracy of RTA among gate is loss. Figure 3.9 shows this relation, i.e. when number of cells is high, clustering starts to outperform non-clustering approach. Here we propose a way to adaptively select when to perform the clustering based on number of cells. We use 7,000 cells (based on results from the graph) as threshold to decide whether to perform clustering or not in this current partition. If the number of cells is higher than threshold, we employ clustering to reduce weighted cutsize.

4. Methodology

4.1 Algorithms

Here we modified one of the state-of-the-art, timing-driven mincut-based global placement algorithm GEO [8] and call it A-GEO as shown in Figure 4.1. The underlined lines are the lines that are modified from the original GEO. Our A-GEO can be used toward any timing-driven mincut-based global placement and some proposed approaches are applicable to any timing driven placement. A-GEO gives a global placement solution for PPR problem. Based on mincut-based global placement [4,13,27,2,5], we recursively bipartition *NL* until m x n tiles are achieved by calling a subroutine called A-GEO-2way. A-GEO-2way is performed

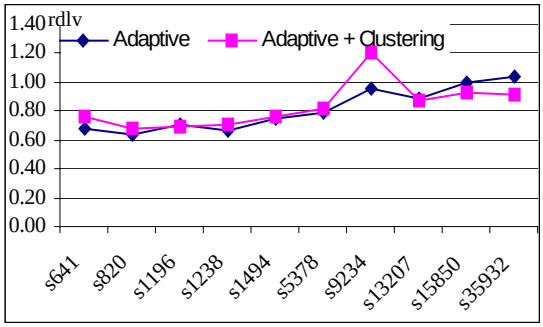

Figure 3.9 Impact on clustering

```
A-GEO(NL,K,run)
1. insert all cells in NL to root node R in T (= partitioning tree)
2. insert R into Q (= FIFO queue)
3. while (leaf nodes in T < K)
4.     N = remove front element in Q
5.     GEO-2way(N,run) (= bipartitioning on N)
6.     split cells in N into N1 and N2
7.    insert N1 and N2 into Q and T
8.    refine
9. return T
-----------------------------------------------------
 A-GEO-2way(N,run)
10. NL' = sub-netlist containing cells in N
11  if  #cells > Threshold T1
12.     ESC(NL') (= multi-level clustering on NL')
13. h = height of the cluster hierarchy
14. B = random partitioning among clusters at level h
15. for (i = h downto 0)
16.     compute adaptive #run
17.     NL'(i) = coarsened NL' at level I
18.     for (j=1 to run)
19.         while (gain)
20.         if not (bottommost level & node id > K/2)
21.         DELAY-WEIGHT(NL'(i))
22.         total net weight = 1 + α delay weight
23.             while (gain)
24.                     move cells in NL'(i) to minimize
                        weighted cutsize
25.             retrieve max gain moves and update B
26.     project best retiming B to level i-1
27. return B
-----------------------------------------------------
 DELAY-WEIGHT(NL')
28. set delay of edges in R (= retiming G)
29. perform RTA(R) (= timing analysis)
30. compute sequential slack for nodes in R
31. for each cluster C in NL'
32.   C(R) = all cells in R grouped into C
33.   slack(C) = min among cells in C(R)
34.   X = ⌈ top x% small slack ⌉ if |X| ≤ 25%
35.   if #cells < threshold T2 use T filter
36.   else A filter
37. for each net N in NL'
38.   if original GEO filter
39.     compute delay-weight(N) using Eqn1
40.   elif (T filter and at least two cells in N are in X)
41.     compute delay-weight(N) using Eqn1
42.   elif (A filter and all clusters in N are in X)
43.     compute delay-weight(N) using Eqn1
-----------------------------------------------------------------
```

Figure 4.1. Overview of the A-GEO algorithm

on the sub-netlist. After all bipartitions are finished for the current level then we perform the refining in that entire level on the sub-netlist. Initially, the partitioning tree *T* has only root node *R*. Then all cells in *NL* are inserted into *R*. The FIFO (First In First Out) queue *Q* is used to support the recursive breadth-first cut sequence.

A-GEO-2way first generates a sub-netlist from the given partition tree node and performs multi-level clustering on it. An ESC clustering algorithm [7] is used for this purpose. Then we obtain a random initial partitioning *B* among the clusters at the top level of the hierarchy. The subsequent top-down multi-level refinement is used to improve *B* in terms of delay. For timing driven global placement, RTA [17] has to be performed to identify timing critical cells/clusters. Then we compute the delay weights for the nets in the sub-netlist for delay optimization. The subsequent iterative improvement through a cluster move tries to minimize the weighted cutsize. Finally the current solution is projected to the next level coarser netlist for multi-level optimization. At the end of A-GEO-2way, two new children nodes are inserted into *T* based on *B*.

The modifications from GEO [8] are as follows: first, we know that selecting from best weighted cutsize does not guarantee best retiming delay among all runs. Instead of returning *B* as best weighted cutsize among all runs, we project *B* as having the best retiming delay among all runs as shown in line 26. Second, we consider adaptive α in line 20, i.e.: if the partitioned node is the bottommost node of the root's right sub-tree then we only minimize cutsize, otherwise we set α=20. Third, we adapt the number of run in line 16: having high number of runs during earlier partitions as described in section 3.3. Here we use 20 runs for the first 3 levels and 3 runs for the rest. Fourth, we employ adaptive filtering based on number of cells in the current partition as in lines 35,36,40 and 42. Fifth, we consider the ε characteristic while select ε value. If circuit characteristic is in slow start mode, we use ε = 5%. On the other hand, If the circuit characteristic is in medium start mode, we use the first value in the medium range as ε, and if it falls in the fast start mode, we set α = 0 and consider only cutsize instead as shown in line 34. Finally, we decide to perform clustering based on the number of cells in line 11.

4.2 Complexity

Since we project best retiming delay to the next level, we then require calculating retiming delay after each run. However RTA itself is based on finding the longest path which requires O(nlog n). To be more precise, our algorithm requires $run \times K \times n \log n$ where run represents number of runs, K represents the number of partitions, and n represents number of cells.

5. Experimental Results

Our algorithms are implemented in C++/STL, compiled with gcc v2.96 with –O3, and run on Pentium III 746 MHz machine. The benchmark set consists of twelve circuits from ISCAS89 [29] and six circuits from ITC99 [28] suites. The statistical information of benchmark circuits is as shown earlier in Table 3.1. We assume unit delay for all gates in the circuits. We report our result in Table 4.1 on 8x8 tiles. GEO represents a state-of-the-art timing driven mincut-based global placement proposed in [8] with five runs. A-GEO represents the modified GEO algorithm with our

Table 4.1 Comparison among GEO, GEO+200r and A-GEO

ckt	GEO		GEO + 200r		A-GEO	
	Dr	wire	Dr	wire	Dr	wire
s641	143	409	137	383	97	442
s820	47	599	42	631	28	582
s1196	74	1,032	74	1,047	49	1,025
s1238	77	1,128	70	1,019	50	1,095
s1494	55	997	51	1,024	41	1,055
s5378	57	1,453	51	1,371	45	1,907
s9234	50	1,459	48	1,408	48	2,132
s13207	86	1,689	72	1,491	69	2,091
s15850	90	1,824	88	1,708	83	2,025
s35932	45	2,113	47	1,903	41	2,536
s38417	41	2,394	39	2,054	41	2,610
s38584	81	3,184	75	2,371	59	4,450
b14o	67	3,658	64	3,704	64	4,114
b15o	79	5,786	72	5,306	79	5,773
b20o	74	6,087	73	5,990	64	7,158
b21o	79	6,149	67	5,775	70	6,941
b22o	80	7,620	63	7,229	63	8,774
Avg.	1.00	1.00	0.93	0.91	0.82	1.14
Time	1,517		51,233		14,232	

adaptive methods with about 4.88 runs (using adaptive number of runs). We also report GEO+200r with is GEO with 200 runs to be fair since our A-GEO has a higher running time than original GEO. The average ratio and running time are also reported and measured in seconds. Results from Table 4.1 shows that the A-GEO is better than the GEO by about 21.9%, and better than the GEO+200r by 13.1%. Note that the GEO+200r requires more running time that the A-GEO by about four times. Hence by increasing number of run alone is not as good as using our adaptive method.

6. Conclusion and Future Work

We propose an adaptive methodology to improve timing driven placement using adaptive parameters. Our method can improve a state-of-the-art timing driven placement GEO [8] by as much as 67% and 22% on average for performance improvement. We are working to employ c-timing [20] instead of retiming to reduce the running time since our algorithm require a lot of retiming calculation and c-timing is faster than retiming.

7. References

[1] C. Ababei, N. Selvakkumaran, K. Bazargan, and G. Karypis, "Multi-objective Circuit Partitioning for Cut size and Path-Based Delay Minimization," *IEEE International Conference in Computer Aided Design*, page 181-185, 2002.

[2] C. J. Alpert and T. F. Chan and A. B. Kahng and I. L. Markov and P. Mulet. Faster Minimization of Linear Wire length for Global Placement. *IEEE Trans on Computer-Aided Design*, 1998.

[3] G. Beraudo and J. Lillis. Timing Optimization of FPGA Placements by Logic Replication. *ACM Design Automation Conf.* page 196-201, 2003.

[4] M. A. Breuer. Class of Min-cut Placement Algorithms. *ACM Design Automation Conf.*, page 284-290, 1997.

[5] A. E. Caldwell and A. B. Kahng and I. L. Markov. Can recursive bisection alone produce routable placements?. *ACM Design Automation Conf.*, page 477-482, 2000.

[6] P. Chong and R. K. Brayton. Characterization of feasibility retimiings. *In Proc. Int. Workshop on Logic and Synthesis*, pages 1-6, 2001.

[7] J. Cong and S. K. Lim, "Edge separability based circuit clustering with application to circuit partitioning," *to appear in IEEE Trans on Computer-Aided Design*, 2003.

[8] J. Cong and S. K. Lim, "Physical Planning with Retiming," *IEEE International Conference in Computer Aided Design*, page 2-7, 2000.

[9] J. Cong and X. Yuan. Multilevel Global Placement with Retiming. *ACM Design Automation Conf.* page 208-213, 2003.

[10] H. Eisenmann and F. M. Johannes. Generic Global Placement and Floorplanning. *ACM Design Automation Conf.*, page 269-274, 1998.

[11] C. Fiduccia and R. Mattheyses, "A Linear Time Heuristic for Improving Network Partitions," *ACM Design Automation Conf.*, page 175-181, 1982.

[12] R. Ho, K. W. Mai and M. A. Horowitz. "The Future of Wires." In *Proceedings of IEEE*, 2001.

[13]D. Huang and A. B. Kahng. Partitioning-based Standard-cell Global Placement with an Exact Objective. Int. Symp. on Physical Design, pages 18-25, 1997.

[14] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel hypergraph partitioning: Application in VLSI domain," *ACM Design Automation Conf.*, page 526-529, 1997.

[15]J. M. Kleinhans and G. Sigl and F. M. Johannes and K. J. Antreich. GORDIAN: VLSI placement by quadratic programming and slicing optimization. *IEEE Trans on Computer-Aided Design*, 1998.

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

[17] C. E. Leiserson and J. B. Saxe, Retiming synchronous circuitry. *Algorithmica*, page 5-35, 1991.

[18] 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

[19] 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.

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

[21] P. Pan, A. K. Karandikar, and C. L. Liu, "Optimal clock period clustering for sequential circuits with retiming," *IEEE Trans on Computer-Aided Design, pages 489-498,1998.*

[22] 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.

[23] SIA, National Technology Roadmap for Semiconductors, 1997.

[24] W. J. Sun and C. Sechen. Efficient and effective placement for very large circuits. *IEEE Trans on Computer-Aided Design,* pages 349-359, 1995.

[25] W. Swartz and C. Sechen. Timing Driven Placement for Large Standard Cell Circuits. *ACM Design Automation Conf.*, page 211-215, 1995.

[26] 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.

[27] J. Vygen. Algorithms for Large Scale Flat Placement. *ACM Design Automation Conf.*, page 746-751, 1997.
[28] http://www.cad.polito.it/tools/itc99.html
[29] http://www.cbl.ncsu.edu