Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 04-040

TheTo — A Fast, Scalable and High-Quality Partitioning Driven Placement Tool.

Navaratnasothie Selvakkumaran and George Karypis

October 26, 2004

THETO — A Fast, Scalable and High-Quality Partitioning Driven Placement Tool.

ABSTRACT

Partitioning driven placement approaches are often preferred for fast and scalable solutions to large placement problems. However, due to the inaccuracy of representing wirelength objective by cut objective the quality of such placements often trails the quality of placements produced by pure wirelength driven placements. In this paper we present THETO, a new partitioning driven placement algorithm that retains the speed associated with traditional partitioning driven placement algorithms but incorporates a number of novel ideas that allows it to produce solutions whose quality is better than those produced by more sophisticated and computationally expensive algorithms. The keys to THETO's success are a new terminal propagation method that allows the partitioner to exactly map the half-perimeter wirelength cost to min-cut cost, and an integral post-bisectioning refinement step that enhances the effectiveness of the new terminal propagation. Experimental validations show that TheTo is fast and also quite effective in reducing half-perimeter wirelength.

Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design Aids

General Terms

Algorithms, Experimentation

Keywords

Partitioning, Placement, wirelength objective

1. INTRODUCTION

Placement is one of the fundamental problems in physical design and numerous algorithms have been developed utilizing a variety of ideas and optimization techniques. However, the ever increasing problem sizes and shortening time-to-market windows require scalable and high-quality solutions in minimal amount of time. These pressures were largely limited to ASIC placement in the past, but modern FPGAs have grown to match the size of ASICs, which necessitates the development of extremely fast global placement algorithms, in order to facilitate reasonable compile times demanded by FPGA users. This has led to the resurgence of partitioning driven placement (PDP) methods, as advances in circuit partitioning resulted in placement algorithms that are computationally scalable, capable of leading to high-quality solutions, and can scale to very large designs. Examples of such partition driven placement tools includes Capo [11] and FengShui [4]. Another tool Dragon [18] uses multilevel partitioning to speed up simulated annealing based hierarchically architected algorithm.

In this paper we present THETO, a new top-down hierarchical partitioning driven placement algorithm that incorporates a number of novel ideas to further improve the quality of the placement solution. Our key contributions are the following: (i) a new method for terminal propagation that takes into account the size of the bounding boxes of the various nets; (ii) a new step in the overall structure of the hierarchical partitioning driven placement framework that further improves the quality of the bisections after the computation of each level; (iii) a comprehensive experimental evaluation of various algorithmic choices for partitioning driven placement and their impact on both quality and computational requirements. Our experimental comparisons against a number of state-of-the-art placement tools shows that THETO outperforms other partitioning-driven-placement schemes by 5%–10%, while having modest computational requirements. Moreover, its overall performance is very competitive against placement schemes that employ either advanced optimization techniques or are based on analytical methods.

The rest of this paper is organized as follows. Section 2 provides some definitions and introduces the notation that is used throughout the paper. Section 3 describes THETO and provides details about the various algorithms that it uses. Section 5 evaluates THETO's performance and compares it against other schemes. Finally, Section 6 provide some concluding remarks.

2. BACKGROUND

The partitioning-driven global placement (*PDP*) paradigm is a divide-and-conquer strategy used to combinatorially partition the netlist and assign the partitions to corresponding geometrically subdivided bins on the two dimensional chip surface. We say this process is applied at multiple levels of global placement, which simply means that we successively solve global placement for finer and finer bin sizes. At the top level, there is only one bin encompassing the entire area of the chip, and all the movable cells of the netlist are at the center of this bin. In the next level, there are two bins which contain bisected portions of the netlist. In the subsequent third level, each of these bins are further subdivided which results

.

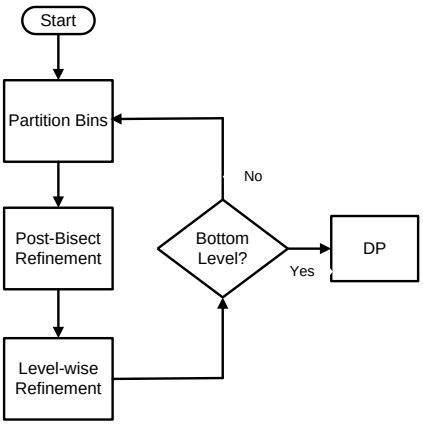

Figure 1: The structure of our PDP algorithm THETO

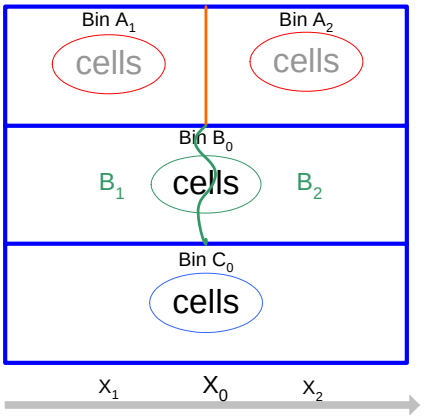

Figure 2: A Snapshot of partitioning the bins

in 4 bins. This process continues until there are $2^m * 2^n$ bins, which is called bottom level.

The detailed placement for standard-cell methodology, essentially takes a global placement input and arranges the cells in rows with minimal perturbation to the wirelength.

A *netlist* $G = (V, E)$ is a set of cells V and a set of nets E. Each net is a subset of the set of cells V. The *size* of a net is the cardinality of this subset. A cell v is said to be *incident* on a net e, if $v \in e$. Each cell v and net e has a weight associated with them and they are denoted by $w(v)$ and $w(e)$, respectively. *Pin* is the location on the cell that physically attaches the net to the cell. The *external portion* of a net is a subnet induced by the cells incident on the net but lie outside the bin currently being considered. Similarly *external cells* of a net are the cells that are incident on the external portion of that net. *Topological neighbors* of a cell v are the subset of cells that are incident on at least one of the nets, on which v is also incident.

The quality of the placement is measured in terms of the half-perimeter wirelength (*HPWL*), which is equal to the weighted sum of half-perimeters of the bounding boxes of the nets that enclose the cells incident on each net, *i.e.*, $\sum half - perimeter(e) * w(e)$. The wirelength objective of the global placement is to minimize HPWL, while satisfying upper-bound and lower-bound constraints on the total weight of cells that each of the bins contains (balance constraint).

3. THE THETO PLACER

Our placement algorithm, called THETO, follows the general top-down hierarchical partition driven placement framework. The overall structure of THETO is shown in Figure 1. It consists two major stages of global placement stage and the detailed placement stage. The global placement stage consists of three distinct steps. The first step computes a bisection of each bin using a cut-based hypergraph partitioning algorithm. The second step, which is applied after each bin has been bisected, further improves the cut (and as a result the wirelength) of the original bisection by taking into account the finer-level partitioning of all the bins. Finally, the last step focuses on minimizing the wirelength of the placed solution at the current level of the hierarchy, by moving/swapping cells between the bins to reduce HPWL. The last step of the THETO is designed with the intention of better addressing global objectives such as timing, however, this step is not necessary for HPWL objective [19].

To a large extent THETO's structure is similar to that of any "textbook" partitioning driven placement algorithm, with the major differences being the use of the post-bisection refinement as one of the integral steps, coupled with a novel terminal propagation scheme that accurately couples HPWL objective of the placement to the min-cut objective of the partitioner. As we will later see in the experimental results section, these schemes significantly improve the quality of the placement and are instrumental in contributing to THETO's overall effectiveness. In the rest of this section we describe various components of THETO.

3.1 Partitioning the bins

THETO bisects each bin using a multilevel hypergraph partitioning algorithm that was derived from hMetis [17]. Multilevel partitioning algorithms are the current state-of-the-art and have been shown to find high-quality partitionings in moderate amount of time. Our locally modified version of hMetis retains its basic overall structure but it has been extended to accept real numbers as net weights and very small balance tolerances. In addition, to further reduce the amount of time spent in partitioning we do not perform any V-cycles and we have reduced the number of coarsening levels that is being computed. In general, the quality of our locally modified version is comparable to that of hMetis 1.5.3, which is available publicly.

3.1.1 Terminal Propagation

Besides the effectiveness of the partitioning algorithm itself, another key factor that determines the overall performance of partitioning driven placement is the method used to take into account, the external portion of nets that participate in any bisection. The goal of this method is to utilize the information that is external to the bin in an effort to bias the min-cut objective toward minimizing HPWL. This technique invented by Dunlop and Kernighan [10], is known as *terminal propagation* (TP).

Traditionally, terminal propagation is performed as follows [9]. For each net that connects internal and external cells and lies exclusively on one half region of the area to be bisected, a fixed dummy cell is added (terminal propagated) to the child bin on that side to try to prevent that net from being extended to the other side (*i.e.*, prevent it from being cut). For computational efficiency [11], instead of assigning a fixed dummy cell to each such net, only two fixed dummy cells are maintained, one for each partition and all the nets that require terminal propagation are attached to these dummy cells. We will refer to this scheme as *traditional terminal propagation*.

During the partitioning process, the weights for the nets are assigned as the same weights of the placement problem. When we try to match the HPWL objective to the mincut objective, this uniform transfer of cost does not reflect the actual situation. Because, in reality the magnitude of HPWL change *does* vary for the nets that

are cut. The decision of whether a net is cut or not, is insufficient to exactly map the change in HPWL. To illustrate this, consider the example shown in Figure 2, which graphically depicts the current state of the bisectioning process of three large bins in a local region of the chip. The top bin has already been bisected into children bins A_1 and A_2, the bottom bin C_0 is yet to be bisected, while the bin in the middle B_0 is currently being bisected. Let the smallest x coordinate of external cells of a net e be less or equal to X_1, and if the largest x of external cells of e is equal to X_1 (connected to some cell in A_1), then if e is cut, it would extend the bounding box by $X_2 - X_1$, but on the other hand if the largest x coordinate of the external cells of e is equal to X_0 (connected to some cell in C_0) then if e is cut, the bounding box will *only* expand by $X_2 - X_0$, which is half as much as $X_2 - X_1$. The traditional terminal propagation scheme does not capture this anomaly in translating HPWL cost to mincut cost, and as a result leads to only an approximate optimization of HPWL objective. However, the terminal propagation is known to be inaccurate in mapping HPWL objective to mincut objective of the partitioner [14, 21].

In the original terminal propagation algorithm [10] and in Capo [11], the terminals that are lying close to the cut-line are ignored. In Capo, this region near the cut line is called "partition fuzziness" region and the suitable range of this region is said to vary from 10%-33% of bin width for different benchmarks [6, 8]. To understand why ignoring the terminals at the cut line creates disparity in objectives, lets consider two nets, a local net e_1 and another one e_2 with a single external terminal at cut line. The best HPWL reduction of e_1 is $X_2 - X_1$, while the best HPWL reduction w.r.t. e_2 is half that much. If both e_1 and e_2 are equally weighted, the partitioning algorithm cannot discriminate between the two and cannot choose e_1 if there is a tie, leading the partitioner to make a sub-optimal choice. The approach of using exact objective [13] is said to overcome the need for any terminal propagation (section4 of [13]). However, the algorithm presented in [13], minimizes the exact cost only if the external terminal that is in the span of bounding box formed by the bin centers, is in alignment with either x or y coordinates of the children bins. For all other terminals (ambiguous terminals), this algorithm [13] is not optimizing the desired objective (such as MST) accurately. However, the algorithm presented by Zhong and Dutt [21], does overcome the need for terminal propagation by explicitly measuring HPWL change for each move. This computationally intensive approach, tries to make computation of bounding boxes more efficient by maintaining binary search trees to store the pins of the nets. Such complex approaches are made redundant by our new terminal propagation scheme, which we describe in detail in the next section.

3.1.2 Bounding Box aware Terminal Propagation

To overcome the deficiency of the traditional terminal propagation, we developed a novel terminal propagation, named as *bounding box aware terminal propagation*(**BBTP**)), that explicitly measures the HPWL change if a net is cut and accurately translates this placement cost to partitioning cost. The partitioning algorithm still minimizes the cut, as they are quite efficient at that, and as the output of the partitioning, we obtain minimal HPWL for that partitioning instance. For any partitioning instance, the lower-bound and the upper-bound for HPWL changes are zero and the half-perimeter of the bounding box formed by the center of bins, respectively. To understand that, consider a local net e in B_0 of Figure 2. Before partitioning, all the incident cells of e are at X_0, thus the initial HPWL cost is zero. After the partitioning, if *all* the incident cells of e are either in B_1 or in B_2, then HPWL cost is still zero. But if the incident cells of e are in both B_1 and B_2, then HPWL cost

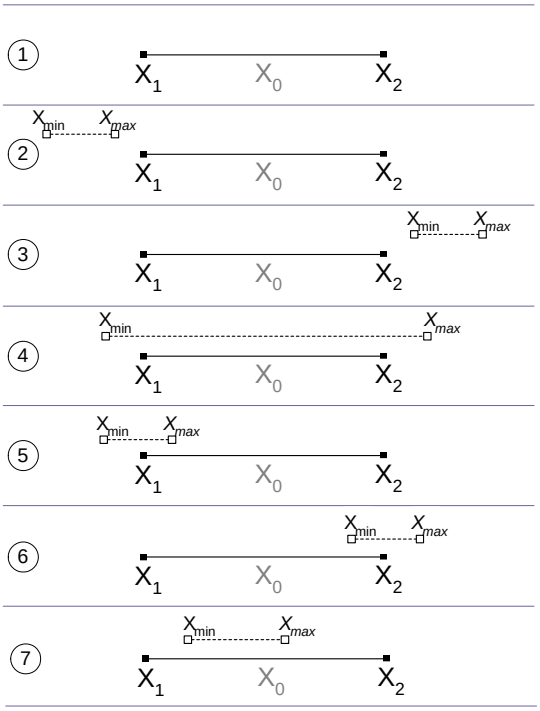

Figure 3: Various scenarios that occur during terminal propagation.

is $X_2 - X_1$. The change in HPWL for the former case is zero and for the latter case it is $X_2 - X_1$. Note that in the example shown in Figure 2 of the bisectioning of bin B_0, the HPWL degradation never exceeds the value of $X_2 - X_1$, as the value of $X_2 - X_1$ is the maximum possible span in x direction for the movement of cells, and the span in y direction is zero. The lower bound (zero) and upper bound ($X_2 - X_1$) of change in HPWL costs are directly translated into the partitioning costs as e being un-cut (zero) and e being cut ($w(e)$), respectively. The traditional terminal propagation algorithm considers *only* these two possibilities and as a result disallows the partitioning algorithm from making more than a binary decision on the cut status of the nets. What our new terminal propagation scheme essentially does is, it allows the partitioner to make "continuous decisions", to account for all the cases where there is change in HPWL and as a result couples HPWL objective with mincut objective exactly.

In order to enable the partitioner to make such "continuous decisions", we multiply the weights of the nets by a re-weighting factor η, which is calculated as the ratio of maximum possible change in HPWL if a net is cut, divided by the value of $X_2 - X_1$. Then we translate the placement cost to partitioning cost, by merely multiplying the net weights by η.

A number of different scenarios arise during the terminal propagation, and these are shown in Figure 3. This figure illustrates the bisectioning process of parent bin B_0, where the cells lying at X_0 are split into two subsets are placed into the child bins B_1 and B_2, such that their x coordinates are equal to X_1 and X_2 respectively ($X_1 < X_2$). The coordinates X_{min} and X_{max} ($X_{min} \leq X_{max}$) denote the minimum and the maximum x coordinates of the cells/pads connected to the current net in consideration, but lie outside B_0 (external terminals).

In the four cases numbered from 1 to 4, presented in Figure 3, the BBTP algorithm is similar to the traditional terminal propagation. First, let us briefly describe these cases before we describe the remaining cases. In case 1, there are no external terminals, thus

there is no need for terminal propagation. In case 2 ($X_{max} \leq X_1$), we need to propagate a terminal to B_1, to discourage the net from being extended to the other side (B_2). A similar scenario, but in the opposite direction is shown in case 3 ($X_{min} \geq X_2$). For these three cases the net re-weighting parameter η is equal to 1.0, as the maximum possible degradation in HPWL is $X_2 - X_1$. For completeness sake, we present the case 4 ($X_{min} \leq X_1$ and $X_{max} \geq X_2$), where irrespective of where the incident cells are placed there is no change in HPWL, and we ignore this net, as in the traditional terminal propagation.

Case 5 ($X_{min} \leq X_1 < X_{max}$) shows where the span of external terminals partially overlap with the span of the centers of bins B_1 and B_2. In this case, if all the internal incident cells of net e lie at X_1, then there is no HPWL degradation, but if any of the cells lie at X_2, then there is HPWL degradation. Therefore, we need to propagate a terminal to B_1, to discourage this net from being extended to bin B_2. The maximum possible change in HPWL for this net is limited to $X_2 - X_{max}$, which is lower than the distance between bin centers ($X_2 - X_1$). In order to communicate this discrepancy to the partitioning algorithm, we need to re-weight the hyperedge of the partitioning problem. For this case the re-weighting factor η is equal to $\frac{(X_2 - X_{max})}{(X_2 - X_1)}$. By following a similar argument, the re-weighting factor η for case 6 ($X_{min} < X_2 \leq X_{max}$) can be identified as $\frac{(X_{min} - X_1)}{(X_2 - X_1)}$.

The most interesting scenario is presented in case 7 ($X_1 < X_{min} \leq X_{max} < X_2$) , in which the span of external terminals is subsumed by the span of bins. As shown in case 7, if all internal cells are at X_1, the HPWL degradation is $(X_{min} - X_1)$, and if all the internal cells are at X_2, then the HPWL degradation is $(X_2 - X_{max})$. One solution for this problem is to choose the location of the terminal based on the magnitude of possible HPWL degradations ($(X_{min} - X_1)$ and $(X_2 - X_{max})$) and propagate a terminal to B_1 if $(X_{min} - X_1)$ is less than $(X_2 - X_{max})$, and propagate a terminal to B_2 if the converse is true. However, this solution results in erroneous mapping of the objectives. To understand that, lets say $(X_{min} - X_1) \leq (X_2 - X_{max})$, and we would propagate a terminal to B_1 for a net e. Now, the only possible gain in HPWL, that the partitioning algorithm aware of is $(X_2 - X_{max})$, as there is always a fixed terminal associated with e in B_1. The partitioning algorithm may or may not leave any of the movable cells incident on e at X_1, as the mincut cost for both choices are same, but if the partitioning algorithm does not leave any movable cells at X_1 a gain of $(X_{min} - X_1)$ in terms of HPWL will result. This results in inaccurate mapping of objectives. To solve this problem, we duplicate the net and attach one copy of the net to the fixed dummy cell in B_1 with the re-weighting factor $\eta = \frac{(X_2 - X_{max})}{(X_2 - X_1)}$, and attach the second copy of the net to the fixed dummy cell in B_2 with the re-weighting factor of $\eta = \frac{(X_{min} - X_1)}{(X_2 - X_1)}$. This ensures the mincut choices made by the partitioning algorithm exactly maps to HPWL changes. Note, if both of these copies of the net are cut, that means the contribution of these nets to the mincut is equivalent to $\frac{(X_2 - X_{max})}{(X_2 - X_1)} + \frac{(X_{min} - X_1)}{(X_2 - X_1)}$, which exactly maps to the HPWL degradation of $(X_2 - X_{max}) + (X_{min} - X_1)$ (as cells lie at both X_1 and X_2).

These seven scenarios presented here covers all possible cases encountered in terminal propagation, and as we have shown here, it is quite possible to exactly map the HPWL to the mincut objective. In our actual implementation, our algorithm can handle the partitioning of diagonal bins too. But for simplicity, we presented the partitioning of horizontal bins only. Furthermore, we have also modified a well-known multilevel partitioning tool to accept the net weights in real numbers. We had to replace the FM bucket data structure with maximum priority queues to satisfy this requirement. Coupled with BBTP that exactly matches HPWL objective to mincut objective, the availability of the partitioner that accepts real numbers for net weights completely satisfy our requirement of minimizing HPWL by minimizing the simpler objective of mincut.

3.2 Post-Bisection Refinement

During the course of bisecting the various bins at each level of the hierarchy, the final locations of the cells at that level are known only for those the bins that have been bisected already. As a result, terminal propagation cannot achieve its full potential in accounting for the external portions of the nets, as the "future locations" for the cells in bins yet to be bisected are unknown. The post-bisection refinement step introduced in THETO is designed to address this problem as it is being applied once all the bins have been bisected and attempts to further improve the quality of each individual bisection and further reducing the HPWL. We have experimented with the delayed V-cycle, in which we delay V-cycle of the bisections till all the bins are bisected. Also, we have used an FM like algorithm that directly minimizes HPWL. In our previous study we found that compared to these two methods, computing an entirely a new bisection and accepting it if the new cut is better than the existing cut (*Repartitioning*) performed better [19]. A parallel but independent study [5] also found similar results in reducing the ambiguity of terminals at the cut line. In this paper, due to the superiority of repartitioning scheme, we limit our discussion of post-bisection refinement to that scheme only. The idea behind repartitioning is well known and it has been widely used in the past [15, 13]. We have experimented with different variations of repartitioning algorithm using a number of different schemes in choosing the pairs of bins for repartitioning. Based on how we chose the pairs of bins, we could categorize the repartitioning schemes into three categories named as "Diagonal", "Connectivity-based" and "Axis Parallel". The schemes of these categories, as the names suggest, pick adjacent bins for repartitioning that are diagonally located, highly connected and parallel to one of the axis, respectively. The "diagonal" scheme is intended to mimic the effect of quadrisectioning, while the "connectivity-based" scheme focuses the partitioning effort on pairs of bins that share most nets between them (Hyperedges are represented as clique of edges and their respective edge weights between bins are summed up). Due to high computational cost, this "connectivity-based" scheme is limited to top-level hierarchy only (less than 1024 bins). We developed six different schemes under "Axis Parallel" category; short descriptions of them follows.

"Hierarchy" - It follows the cut line hierarchy, and redo the partitioning performed during last bisectioning. (we call this cut line as last-bisection cut line).

"Orthogonal" - This scheme chooses bins such that the repartitioning straddles across an ancestral cut line, that is orthogonal to the last-bisection cut line.

"Parallel" - Repartitioning straddles across an ancestral cut line that is parallel to last-bisection cut line.

"X-Y" - This scheme chose the direction based on relative components (horizontal, vertical) of the interconnect distribution. Specifically, if the horizontal wirelength is more, we pick horizontally adjacent bins, otherwise we pick vertically adjacent bins.

"all" - In this scheme we apply all the above for axis parallel schemes.

"all-perturb" - In this scheme, we perturb the solution slightly such that the cells are moved away from bin centers. Motivated by the SOR algorithm presented in [8], we force the cells to descend towards their center of gravity locations by small distances. We repeat this for couple of iterations.

The relative performances of all these schemes are presented in the experimental section Section 5.1.2.

4. DETAILED PLACEMENT

Our detailed placement algorithm has two major components. The first one is an adaptation of recently described dynamic programming based legalization algorithm [4], and the second component is a commonly used window based permutation of cells [18].

4.1 Dynamic Programming based Legalization

A number of detailed placement algorithms rely on dynamic programming formulation [4, 20, 16]. We adapt an algorithm recently developed by Agnihotri et.al. ([4]), by replacing the objective of minimum squared move distance with the objective of minimum HPWL. Essentially, we select more candidates than necessary to fill the width of the row (typically about 3-5 times the row width), and by using the dynamic programming formulation we decide which cells are preferable for assignment and which cells are preferable for deferment to the next row. Due to page limitations we do not elaborate this algorithm further, for more information please refer [4].

One of the disadvantages of this formulation is the restriction placed on the order of cells (in x direction) to obey the original order of the candidates. We experimented with extensions that partially remove these restrictions, but found the basic formulation coupled with the randomized window based permutation refinement provided the best trade-off in terms of runtime and quality.

After the dynamic programming based refinement , we also perform greedy swaps of cells with their topological neighbors if both the cells are of same size. By limiting the neighbors to be of same size, the legality of the placement is preserved. This swapping coupled with an iteration of randomized window based permutation of cells forms the next step of our detailed placement.

4.1.1 Randomized Window based Permutation of Cells

One commonly used method in detailed placement is computing the permutation of cells to change their order in x direction [18]. For this purpose, we randomly identify a window of 5-7 consecutive cells in a row and then exhaustively compute the wirelength cost of all possible permutations of these cells. If the total width of cells inside a window exceeds the capacity of the window, then we expand the possible x coordinates of the window in both left direction and right direction. Whenever, there are white-spaces among such selected cells, we treat each white space (i.e. width of single site) as individual pseudo cell, and compute the permutation of both types of cells together. We use each cell of a particular row as an anchor point (center cell) for the randomly selected window, thus guaranteeing all cells are visited at least once. We repeatedly apply this step to the rows in a randomly selected order for a few iterations. Note that, we do not use a sliding window based optimization, as it favors the cells that need to move in the direction of the sliding window, and disfavors the cells that need to move in the opposite direction. Randomly choosing the windows eliminates such biases.

5. EXPERIMENTAL VALIDATION

We evaluated the performance of the various algorithmic choices in THETO on first five circuits chosen from three different benchmark suites, IBM-PLACE v1.0, Peko-Suite-1 and IBM-PLACE v2.0 (Easy). The first and the third benchmarks were obtained from [1], while the second one was obtained from [2]. (Note that IBM-PLACE benchmarks have been re-labeled as IBM-DRAGON benchmarks in some publications). The characteristics of these benchmarks are listed in Table 1. We performed all our experiments on 1.5GHz Athlon MP machines with 2GB of memory and we compiled our programs with aggressive optimization settings (gcc v3.3 -O3 -ffast-math -funroll-all-loops -fomit-frame-pointer).

name	num cells	num nets
ibm01	12282	11507
ibm02	19321	18429
ibm03	22207	21621
ibm04	26633	26163
ibm05	29347	28446
peko01	12994	13865
peko02	19950	19325
peko03	23513	27118

name	num cells	num nets
peko04	27938	31683
peko05	28878	27777
ibm01-cu85	12274	11507
ibm02-cu90	19321	18429
ibm07-cu90	45098	44394
ibm08-cu90	50958	47944
ibm09-cu90	51667	50393

Table 1: The benchmark circuits.

The performance of the various schemes was evaluated by comparing two quantities. The first is the half-perimeter wirelength (denoted as "HPWL" in the tables), which measures the quality of the solution in million units. Solutions that have smaller HPWL values are better. The second is the amount of time required to compute the solution (denoted as "Time" in the tables). Schemes that require less time are preferred over those requiring more time. The numbers presented are average results of 5 independent runs. We present run time in cpu seconds unless noted otherwise. Also, to make overall comparisons between different schemes across the different data sets easier, we compute two *summary* statistics. The first is the total amount of time (denoted as "TTime" in the tables), which is simply the time required to place all 15 benchmarks. The second is called *average quality relative to the best* (denoted "AQB" in the tables) and measures the relative performance of the various schemes being compared in terms of HPWL. The AQB statistic for a particular scheme is computed as follows. For each benchmark we compute the ratio of the HPWL produced by that scheme against the smallest HPWL produced for that benchmark by any of the schemes under consideration, and we obtain its AQB by taking the geometric mean of these ratios across the 15 benchmarks. A scheme that achieved an AQB value that is 1.0 means that for all benchmarks it produced the smallest HPWL. In general, a scheme will outperform another, if its AQB value is smaller. AQB-GP refers to global placement result and AQB-DP refers to detail placement result. We used "RowIroning Utility" obtained from [3] to validate the legality of our detailed placement results.

5.1 Evaluation of Various Algorithmic Choices

In this section we provide empirical analysis of various options available in THETO. When we discuss global placement algorithms, we limit our presentation to global placement results obtained by recursively bisecting till there are $16 * |V|$ bins. Even though this placement is very close to detailed placement, overlaps exist, and we report the detailed placement results where relevant.

5.1.1 Terminal Propagation Schemes

The performance achieved by the different terminal propagation schemes described in Section 3.1.1 is shown in Table 2. Specifically, this table shows the performance achieved by four different schemes. The first two schemes use the traditional terminal propagation scheme (labeled "Traditional TP"), whereas the other two are based on the new bounding box aware scheme (labeled "BBTP"). The difference between each pair of schemes is the number of different bisections that they compute during each bin-bisection step. In particular, the schemes labeled "NRuns=1" compute a single bisection, whereas the schemes labeled "NRuns=5" compute five different bisections and select the one that achieves the smallest cut. These results were obtained without performing any post-

	Traditional TP				BBTP			
	NRuns= 1		NRuns=5		NRuns=1		NRuns=5	
	HPWL	Time	HPWL	Time	HPWL	Time	HPWL	Time
ibm01	5.85	7	5.38	19	5.39	9	5.13	22
ibm02	16.53	13	15.73	33	14.88	16	14.54	40
ibm03	15.31	14	14.45	36	14.09	18	13.48	42
ibm04	20.27	18	18.76	44	18.68	22	17.63	51
ibm05	47.41	20	41.61	50	42.85	25	39.14	61
peko01	1.69	8	1.52	19	1.55	10	1.35	22
peko02	2.88	14	2.42	33	2.54	17	2.09	38
peko03	3.21	16	2.85	38	2.94	19	2.51	43
peko04	3.62	19	3.38	46	3.47	23	2.96	51
peko05	4.51	20	3.72	49	3.94	25	3.28	56
ibm01-cu85	57.79	8	55.50	23	53.70	9	52.50	26
ibm02-cu90	163.35	15	158.55	41	149.86	17	146.78	49
ibm07-cu90	369.23	41	346.98	120	343.92	49	326.18	136
ibm08-cu90	425.20	50	401.58	144	370.55	57	358.98	166
ibm09-cu90	344.88	48	324.71	141	315.04	53	303.58	155
AQB-GP	1.193		1.094		1.088		1.000	
TTime-GP		311		837		368		956

Table 2: Results obtained by terminal propagation schemes.

bisection refinement.

¿From these results we can see that the new terminal propagation scheme is superior to the traditional approach as it leads to higher quality solutions without substantially increasing the overall run time. For example, when NRuns=1, BBTP leads to solutions that have 10.5% lower global placement HPWL, while incurring only a 20% increase in runtime. Similarly, when NRuns=5, the solution produced by BBTP is 9.4% better over the solution produced after five runs of the traditional TP. Comparing the impact of improved bisectioning quality (that is achieved by increasing the number of runs), we can see that it directly translates to lower HPWL. For example, for NRuns=5, the results of the traditional TP and the BBTP improved on the average by 9.9% and 8.8%, respectively. However, it is interesting to note that for this set of benchmarks, a single run of the BBTP algorithm produces lower HPWL than five runs of the traditional TP scheme, even though the latter requires 2.2 times more time. For the rest of the paper, we will refer to the results obtained from the five runs of the BBTP scheme as BBTP5 and use it to establish a performance baseline in order to facilitate comparisons across different results.

5.1.2 Various Repartitioning Schemes

In Table 3, we present the impact of various repartitioning schemes described in Section 3.2. In the second column, we show the results of bisectioning without any repartitioning. This result was obtained by using NRuns=1 for finer level bins and NRuns=3 for the first 16 bisections. This particular setting results in only about 3% worse results than BBTP5, but it is twice as faster. The "Axis Parallel" schemes are presented in columns 3-8, while columns 9,10 and 11 present the results of "diagonal", "connectivity" based repartitioning and the results of BBTP5 respectively. Among these schemes both "hierarchy" and "X-Y" methods perform equally well and about 2% better than the schemes that repartition across the ancestral cut lines. The scheme "all" performs better than these two as expected (more coverage and more effort). The scheme "all-perturb", had turned out to be the overall winner by out-performing all other repartitioning schemes. The "connectivity" and "diagonal" schemes improve the results slightly (about 1%) but they are inferior to "axis parallel" schemes.

5.2 Distribution of X–Y Components of HPWL

To evaluate the congestion characteristics of the placements produced by THETO we looked the X and Y components of the wirelength of the various nets and considered the extent to which the solution leads to placements that balances these components. The motivation behind this evaluation metric is, assuming the availability of equal routing resources in both directions, a placement solution that equally distributes the wirelength aong the two directions will have lower congestion over a solution that tends to prefer one direction over the other.

Table 4 shows the relative distributions of horizontal and vertical interconnects for THETO's global and detailed placement and compares it against the detailed placement solution produced by Capo 8.8. These results show that THETO's global placement solution achieves relatively well-balanced interconnect distribution in terms of both horizontal and vertical components. The unbalance percentage (UB%) of these two components is on average about 4.5%, while the maximum unbalance percentage between these components is 14% (seen in ibm04). However, when THETO's detailed placement is considered, we can see that the overall balance becomes somewhat worse. We believe that the reason for this degradation is the fact that the detailed placement algorithm does not currently perform any flipping or rotation to improve the X component. Consequently, the X component tends to increase as we go from global placement to detailed placement. Nevertheless, THETO achieves much better distribution compared to the distribution seen in the results of Capo 8.8. The balancing of these components in THETO is primarily due to the alternating cut directions during global placement. Also note that the "X-Y" repartitioning step (Section 5.1.2), does improve the overall balance but only by a marginal amount (about 1%).

5.3 Comparison with Other Placement Tools

Table 5 provides the overall performance of THETO w.r.t. a number of recent academic placement tools. In this table, we present the detailed placement wirelength and complete run times of tools Dragon [18], Capo 8.8 [11], FengShui2.2 [4][1], mPL (both the latest published results of version 2 - mPL2 [12], and the latest release version 4 - mPL4). The results for mPL2 were obtained from [7], whereas in order to facilitate proper runtime comparisons, for the rest of the schemes, we computed the placements locally using the publicly available versions of the respective algorithms. Finally, the results for THETO correspond to the detailed placement solution of

[1]We do not use most recent version of 2.6 as it is said to produce erroneous results.

	none	Axis Parallel Repartitioning						Diagonal	Connectivity	BBTP5
circuit		hierarchy	orthogonal	parallel	X-Y	all	all-perturb			
ibm01	5.18	5.10	5.21	5.15	5.08	5.01	4.97	5.22	5.19	5.13
ibm02	14.76	14.30	14.31	14.53	14.42	13.77	13.90	14.54	14.50	14.54
ibm03	13.65	13.41	13.47	13.66	13.33	13.09	12.89	13.72	13.67	13.48
ibm04	18.20	17.46	17.76	17.84	17.63	17.27	17.06	17.90	17.83	17.63
ibm05	39.52	38.28	38.55	38.45	38.78	37.37	37.31	39.25	38.83	39.14
peko01	1.40	1.32	1.42	1.43	1.34	1.31	1.28	1.41	1.40	1.35
peko02	2.24	2.09	2.16	2.20	2.11	2.00	1.98	2.25	2.20	2.09
peko03	2.65	2.57	2.61	2.62	2.59	2.47	2.37	2.64	2.57	2.51
peko04	3.11	2.89	3.08	2.99	2.85	2.86	2.69	3.10	3.00	2.96
peko05	3.45	3.17	3.24	3.38	3.16	3.08	2.90	3.30	3.27	3.28
ibm01-cu85	53.25	51.54	52.10	52.52	52.53	51.18	51.27	52.68	52.88	52.50
ibm02-cu90	148.23	144.50	145.28	146.22	145.88	141.54	140.83	147.64	148.44	146.78
ibm07-cu90	332.89	323.36	321.49	324.31	320.72	312.87	311.53	325.60	327.14	326.18
ibm08-cu90	370.06	352.88	361.98	355.33	359.39	344.90	338.85	359.75	368.95	358.98
ibm09-cu90	312.70	298.56	300.55	302.61	297.88	290.84	291.27	301.32	304.89	303.58
AQB-GP	1.087	1.042	1.063	1.068	1.047	1.017	1.001	1.075	1.070	1.056
Time-GP	475	708	634	696	702	1321	1367	767	616	960

Table 3: Impact of using various repartitioning schemes.

	Repart-All-Perturb						Capo 8.8		
	GP			DP			DP		
circuit	x	y	UB%	x	y	UB%	x	y	UB%
ibm01	0.45	0.55	10	0.43	0.57	14	0.60	0.40	20
ibm02	0.49	0.51	2	0.49	0.51	2	0.65	0.35	31
ibm03	0.5	0.5	0	0.5	0.5	0	0.56	0.44	13
ibm04	0.43	0.57	14	0.45	0.55	10	0.63	0.37	26
ibm05	0.52	0.48	4	0.52	0.48	4	0.63	0.37	26
peko01	0.51	0.49	2	0.55	0.45	10	0.58	0.42	15
peko02	0.5	0.5	0	0.56	0.44	12	0.58	0.42	16
peko03	0.48	0.52	4	0.55	0.45	10	0.57	0.43	15
peko04	0.5	0.5	0	0.56	0.44	12	0.59	0.41	18
peko05	0.5	0.5	0	0.56	0.44	12	0.58	0.42	16
ibm01-cu85	0.5	0.5	0	0.55	0.45	10	0.47	0.53	5
ibm02-cu90	0.54	0.46	8	0.55	0.45	10	0.51	0.49	1
ibm07-cu90	0.55	0.45	10	0.58	0.42	16	0.53	0.47	6
ibm08-cu90	0.52	0.48	4	0.55	0.45	10	0.53	0.47	7
ibm09-cu90	0.55	0.45	10	0.58	0.42	16	0.48	0.52	5
AVG-UB%			4.5			9.9			14.6
MAX-UB%			14			16			31

Table 4: X-Y interconnect distribution of THETO and Capo 8.8.

the "all-perturb" results of Table 3.

A number of key observations can be made from these results. First, comparing THETO against the other two partitioning-driven-placement schemes (i.e., Capo and FengShui), we can see that THETO produces solutions that on the average have smaller HPWL than either one of them. The quality advantage over Capo is around 9.6%, whereas the quality advantage over FengShui is 2.6%. In terms of runtime, THETO is about 14% slower than Capo and 2.28 times faster than FengShui. Comparing THETO against Dragon, we see that it has a significant runtime advantage, as it is more than an order of magnitude faster, whereas at the same time it produces solutions whose HPWL is 5.7% better. However, the performance advantage of THETO against mPL is mixed. Specifically, THETO produces solutions whose HPWL is 3% better over that of mPL2, but it is 10.9% worse against the most recent release mPL4 (which became available as we were writing this manuscript). We believe that the reason for this performance difference is the fact that THETO's DP algorithm is not fully optimized, and that there is room for additional quality gains. However, even against mPL4, THETO has the advantage of being more than twice as fast.

6. CONCLUSION

In this paper we presented THETO a new placement algorithm based on partitioning driven placement. The key contributions of this work was (i) the introduction of the bounding-box aware terminal propagation scheme, which is able to correctly translate the HPWL objective to the min-cut partitioning objective, and (ii) the development of new post-bisection refinement strategies. Our comprehensive experimental evaluation showed that the combination of these two schemes can considerably improve the quality of the solutions produced by the PDP framework. In terms of quality, THETO outperforms other state-of-the-art PDP- and non-PDP-based schemes by 5%-10%, while having modest computational requirements.

Our research thus far has focused on developing new schemes and methodologies to improve the global placement phase. However, we believe that THETO can be further improved and we are currently investigating methods to better integrate the detailed placement phase within the multilevel framework used during global placement.

7. REFERENCES

[1] http://er.cs.ucla.edu/benchmarks/ibm-place/index.html.

[2] http://ballade.cs.ucla.edu/ pubbench/peko.htm.

[3] http://vlsicad.eecs.umich.edu/BK/PlaceUtils/.

[4] A.Agnihotri, M.C.Yildiz, A.Khatkhate, A.Mathur, S.Ono, and P.H.Madden. Fractional cut : Improved recursive bisection placement. In *Proc. of ICCAD*, pages 307–310, 2003.

circuit	Dragon-3.01	Capo8.8	Fengshui22	mPL2	mPL4	THETO
ibm01	4.55	5.00	4.91	5.30	4.81	5.32
ibm02	13.35	15.69	14.33	14.90	13.54	14.58
ibm03	12.53	13.52	12.93	14.80	12.99	13.74
ibm04	15.88	17.88	17.07	18.90	16.51	18.20
ibm05	36.81	43.53	37.53	42.70	38.33	38.41
Peko01	1.62	1.44	1.37	1.10	1.01	1.26
Peko02	2.44	2.25	2.33	1.76	1.56	1.92
Peko03	3.02	2.80	2.90	2.05	1.88	2.36
Peko04	3.94	3.18	3.07	2.31	2.27	2.73
Peko05	4.04	3.53	3.17	2.57	2.40	2.88
ibm01-cu85	51.05	56.13	52.11	64.00	50.80	54.52
ibm02-cu90	144.90	160.00	147.64	161.00	143.57	147.51
ibm07-cu90	333.92	364.03	323.62	407.00	322.39	335.52
ibm08-cu90	343.32	385.06	359.17	425.00	356.01	360.85
ibm09-cu90	298.76	326.79	302.76	381.00	299.44	320.02
AQB	1.181	1.220	1.150	1.154	1.015	1.124
Ttime	43328	2985	7777	n/a	7231	3405

Table 5: Comparison with Other Placement Tools.

[5] A.B.Kahng and S.Reda. Placement feedback: A concept and method for better min-cut placements. In *Proc. of DAC*, pages 357–362, 2004.

[6] S. Adya, I. Markov, and P. Villarrubia. On whitespace and stability in mixed-size placement. In *Proc. of ICCAD*, pages 311–317, 2003.

[7] S. Adya, M. Yildiz, I. Markov, P. Villarrubia, P. Parakh, and P. Madden. Benchmarking for large-scale placement and beyond. In *Proc. of ISPD*, pages 95–103, 2003.

[8] S. N. Adya and I. L. Markov. Improving min-cut placement for vlsi using analytical techniques. In *Proc. IBM ACAS Conference*, pages 55–62. IBM ARL, February 2003.

[9] A.E.Caldwell, A.B.Kahng, and I.L.Markov. Optimal partitioners and end-case placers for top-down placement. In *Proceedings of ISPD*, pages 90–96, 1999.

[10] A.E.Dunlop and B.W.Kernighan. A procedure for placement of standard-cell vlsi circuits. In *IEEE Trans. on CAD of Integrated Circuits and Systems*, pages CAD–4(1):92–98, 1985.

[11] A. Caldwell, A.B.Kahng, and I.L.Markov. Can recursive bisection alone produce routable placements? In *Proc. of DAC*, pages 477–482, 2000.

[12] T. Chan, J. Cong, J.Shinnerl, and K. Sze. An enhanced multilevel algorithm for circuit placement. In *Proc. of ICCAD*, pages 299–306, 2003.

[13] D.J-H.Huang and A.B.Kahng. Partitioning-based standard-cell global placement with an exact objective. In *Proc. ISPD*, pages 18–25, 1997.

[14] J.Cong, T.Kong, J.R.Shinnerl, M.Xie, and X.Yuan. Large-scale circuit placement: Gap and promise. In *Proc. of ICCAD*, pages 883–890, 2003.

[15] J.M.Kleinhans, G.Sigl, F.M.Johannes, and K.J.Antreich. Gordian: Vlsi placement by quadratic programming and slicing optimization. In *IEEE Trans. on CAD, Vol-10, No-3*, pages 356–365, 1991.

[16] A. Kahng, P. Tucker, and A. Zelikovsky. Optimization of linear placements for wirelength minimization with free sites. In *Proc. of ASPDAC*, pages 241–244, 1999.

[17] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: Application in vlsi domain. *IEEE Transactions on VLSI Systems*, 20(1), 1999. A short version appears in the proceedings of DAC 1997.

[18] X. M.Wang and M.Sarrafzadeh. Dragon2000: Standard-cell placement tool for large industry circuits. In *Proc. of ICCAD*, pages 160–163, 2000.

[19] N. Selvakkumaran and G. Karypis. Theto - a fast and high-quality partitioning driven global placer. Technical Report Technical Report 03-46, Dept of Computer Science and Engineering, University of Minnesota, November 2003. http://www.cs.umn.edu/tech_reports/.

[20] U.Brenner and J.Vygen. Faster optimal single-row placement with fixed ordering. In *Proc. of DATE*, pages 117–121, 2000.

[21] K. Zhong and S. Dutt. Effective partition-driven placement with simultaneous level processing and global net views. In *Proc. of ICCAD*, pages 171–176, 2000.