An Analytic Placer for Mixed-Size Placement and Timing-Driven Placement*

Andrew B. Kahng and Qinke Wang

Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093-0114, USA
E-mail: {abk,qiwang}@cs.ucsd.edu

Abstract

We extend the APlace wirelength-driven standard-cell analytic placement framework of [21] to address timing-driven and mixed-size ("boulders and dust") placement. Compared with timing-driven industry tools, evaluated by commercial detailed routing and STA, we achieve an average of 8.4% reduction in cycle time and 7.5% reduction in wirelength for a set of six industry testcases. For mixed-size placement, we achieve an average of 4% wirelength reduction on ISPD02 mixed-size placement benchmarks [18] compared to results of the leading-edge solver, Feng Shui (v2.4) [25]. We are currently evaluating our placer on industry testcases that combine the challenges of timing constraints, large instance sizes, and embedded blocks (both fixed and unfixed).

1 Introduction

As VLSI designs scale to billion-transistor complexities, dual challenges arise for the placement phases of the implementation flow. First, design productivity increasingly requires the reuse of predesigned or generated macro blocks (processing and interface cores, embedded memories, etc.). This presents a "boulders and dust" challenge to placers [4], where the sizes of placeable objects can vary by factors of 10,000 or more. Traditional standard-cell frameworks often cannot address this challenge smoothly, and must resort to devices such as manual preplacement of blocks, with an attendant loss of overall solution quality. Second, design value depends on performance; with device and interconnect scaling, this presents greater challenges to timing-driven placement. Motivated by these challenges, in this work we study the two problems of mixed-size placement and timing-driven placement. Specifically, recent work has implemented *APlace*, an analytic placer for wirelength-driven standard-cell placement [21], and obtained superior results compared to a leading industry placer, Cadence QPlace (SE v5.4), as well as recent academic tools: UCLA Dragon (v3.01) [13] and Capo (v8.7) [9]. This paper describes extensions of the APlace tool to handle mixed-size placement and timing-driven placement. Key achievements are:

- Our extension of APlace to mixed-size placement is compared to recent academic tools: UCLA mPG-MS [10], Feng Shui (v2.4) [25] and a three stage placement-floorplanning-placement flow that uses Capo [1] [2]. The half-perimeter wirelength of our placer outperforms that of mPG-MS, Feng Shui and the Capo flow respectively by 24.7%, 4.0% and 26.0% on average.
- Our extension of timing-driven placement is compared to two industry placers: QPlace (SE v5.4) and amoebaPlace (SoC Encounter v3.2). When timing-driven placement is performed for six industry circuits and placements are detail-routed using Cadence WarpRoute (SoC Encounter v3.2), our placer has a minimum cycle time that outperforms that of QPlace and amoebaPlace respectively by 9.6% and 8.5%, as well as average improvements of 7.2% and 6.5% in routed wirelength, respectively.

The remainder of this paper is organized as follows. Section 2 introduces the formulation and implementation of APlace for standard-cell placement. Sections 3 and 4 extend the placer to address mixed-size placement and timing-driven placement. The paper concludes in Section 5.

2 APlace: Analytic Standard-Cell Placement

In this section, we review the formulation and implementation of the APlace analytic standard-cell placer described in [21]. Our present work, which we describe in Sections 3 and 4 below, is embodied in extensions to this tool.

2.1 Problem Formulation

A basic goal of placement is to minimize wirelength subject to the constraint that cells do not overlap. Therefore, the objective function for analytic placement historically includes two terms: a *density objective* to spread cells, and a *wirelength objective* to minimize wirelength.

Cell Spreading. To distribute cells evenly over the placement area, a generic strategy is to divide the placement region into grids and then attempt to equalize the total cell area in every grid. The straightforward "squared deviation" penalty for uneven cell distribution

$$Penalty = \sum_{Grid\ g} (TotalCellArea(g) - AverageCellArea)^2 \quad (1)$$

is neither smooth nor differentiable, and is hence difficult to optimize. The authors of APlace implement the technique of Naylor et al. [31], who smooth this penalty function with a "bell-shaped" cell potential function. For a cell c with center at $(CellX, CellY)$ and

*Work partially supported by the MARCO Gigascale Silicon Research Center, NSF MIP-9987678 and the Semiconductor Research Corporation.

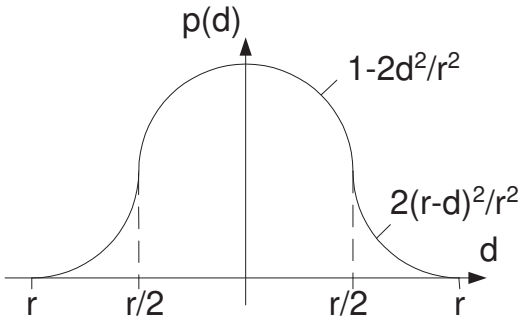

Figure 1: The bell-shaped function.

area A, the potential with respect to grid point $g = (GridX, GridY)$ is given by

$$\begin{aligned} &Potential(c,g) = \\ &C \cdot p(|CellX - GridX|) \cdot p(|CellY - GridY|) \end{aligned} \quad (2)$$

where

$$p(d) = \begin{cases} 1 - 2d^2/r^2 & (0 \le d \le r/2) \\ 2(d-r)^2/r^2 & (r/2 \le d \le r) \end{cases} \quad (3)$$

Here, $p(d)$ defines the bell-shaped function, which is illustrated in Figure 1; r controls the *radius* of any given cell's potential (range of interaction); and C is a normalization factor so that $\sum_g Potential(c,g) = A$, i.e., each cell has a total potential equal to its area. Then, the penalty function in Equation (1) is transformed to:

$$Penalty = \sum_{Grid\ g} \Big(\sum_{Cell\ c} Potential(c,g) - ExpPotential(g) \Big)^2 \quad (4)$$

where $ExpPotential(g) = TotalCellArea/NumGrids$ is the expected total potential at the grid point g.

[21] also integrates congestion information into the objective, via the Kahng-Xu bend-based congestion estimation method [22]. If a particular grid is determined to be congested (resp. uncongested), the expected total cell potential of the grid in Equation (4) is reduced (resp. increased) accordingly. The sum of expected area potential over all grids is kept constant, and equal to the total cell area. Specifically, expected cell potential is adjusted as follows:

$$ExpPotential(g) \propto 1 + \gamma \left(1 - 2 \frac{Congestion(g)}{\max_g \{Congestion(g)\}} \right) \quad (5)$$

where γ is the congestion adjustment factor and decides the relative importance of congestion-directed placement.

Wirelength Minimization. While wirelength and overall placement quality is typically evaluated according to half-perimeter wirelength (HPWL), this "linear wirelength" function can not be efficiently minimized. Nonlinear approximations of HPWL are well-studied in such works as [3] [5] [23] [24]. The approach of Naylor et al. [31], which is implemented in APlace, follows along similar lines, and uses a log-sum-exp method to capture the linear half-perimeter wirelength while simultaneously obtaining the desirable characteristic of continuous differentiability. The log-sum-exp formula picks the most dominant terms among pin coordinates. For a net t with pin coordinates $\{(x_1,y_1), (x_2,y_2), \ldots (x_n,y_n)\}$, the wirelength objective is

$$\begin{aligned} WL(t) = \quad & \alpha \cdot (ln(\sum e^{x_i/\alpha}) + ln(\sum e^{-x_i/\alpha})) + \\ & \alpha \cdot (ln(\sum e^{y_i/\alpha}) + ln(\sum e^{-y_i/\alpha})) \end{aligned} \quad (6)$$

where α is a smoothing parameter.

The APlace analytic placer combines the two objectives and optimizes the function

$$WLWeight * TotalWL + DensityWeight * DensityPenalty \quad (7)$$

Here, the density term drives cell spreading and changes with the current cell distribution. The wirelength term draws connected components back toward each other.

2.2 Implementation

APlace uses the conjugate gradient method to optimize the objective function of Equation (7). A detailed treatment of the conjugate gradient method, along with a survey of descent-based methods for nonlinear programming, can be found in [29]. Control parameters within APlace are weights of the wirelength and density objectives. Intuitively, a larger wirelength weight will draw cells together and prevent them from spreading out, while a larger density penalty weight will spread the cells out (without attention to wirelength). These controls are managed by keeping the density weight fixed at some constant, and setting the wirelength weight to be large at the outset, but then decreasing this weight (by a factor of two, or a smaller factor near the final placement) whenever the conjugate gradient optimizer slows down and a stable solution emerges. After every weight change, the conjugate gradient optimizer is used to compute a new stable state wherein cells are distributed more evenly but wirelength is larger.[1] Results reported in [21] for wirelength and runtime, as well as extensibility to such contexts as area-array IO co-placement, mixed-signal constrained placement, etc. suggest that APlace is a strong and flexible platform for further placement innovations. In the same spirit as recent works that have built on the Capo top-down bisection tool, we now build on APlace.

3 Mixed-Size Placement

As noted above, IP blocks and embedded memories present large or medium-sized pre-designed black boxes that must be placed simultaneously with up to millions of small standard cells. In this section, we extend the APlace approach to address the mixed-size placement problem. Our focus is on two issues: (1) the cell-spreading potential function, and (2) legalization.

3.1 Previous Work

Modern ASIC designs are typically laid out in the fixed-die context, where the outline of the core area, as well as routing tracks and power/ground distribution, are fixed before placement [7]. Mixed-size placement becomes particularly complex in the fixed-die context because of its discreteness [2]. Analytical mixed-size placers have been previously reported in [14] and other works, but have typically been validated on small variable-die layouts, and often place macros with significant overlap.

Recently, a three stage placement-floorplanning-placement flow [1] [2] has been proposed which places macro blocks and standard cells without overlap. In the first step, all macros are shredded into small pieces connected by fake wires. A standard-cell placer, Capo, is used to obtain an initial placement, and the initial locations of

[1] Another aspect of the APlace tool is its integration of top-down hierarchical processing according to the clustering hierarchy computed by MLPart (v4.21), the open-source min-cut hypergraph/circuit partitioner of [8]. For each level in the cell/cluster hierarchy, a coarse grid is determined by the average cluster size. To compute the density penalty, a cluster is essentially regarded as a macro cell with area equal to its total cell area; to compute the wirelength estimate, every cell is assumed to be located at the center of its cluster. The multilevel technique affords additional scalability.

macros are produced by averaging the locations of faked cells created during the shredding process. In the second step, the standard cells are merged into soft blocks, and a fixed-outline floorplanner, Parquet, generates valid locations of macros and soft blocks of movable cells. Finally, with macros considered fixed, Capo is used again to re-place small cells. This approach scales reasonably well, but wirelength results are often quite suboptimal.

A different approach is pursued by [10], wherein a simulated annealing based multi-level placer, mPG-MS, recursively clusters both macro blocks and standard cells to build a hierarchy. The top-level netlist is placed, and then the placement is gradually refined by unclustering the netlist and improving the placement of smaller clusters by simulated annealing. Large objects are gradually fixed without overlap during coarse placement, and the locations of smaller objects are determined during further refinement. Significant effort is needed for legalization and overlap removal during this placement process.

Another recursive bisection based placement tool, Feng Shui, has been presented more recently in [25]. Rather than address standard cells and macro blocks separately, the placer considers them simultaneously via a *fractional cut* technique which allows horizontal cut lines that are not aligned with row boundaries. When compared to previous academic tools, the Feng Shui placer achieves surprising solution quality (as evaluated by placed half-perimeter wirelength) and good scalability.

3.2 Potential Function for Mixed-Size Placement

As described in Section 2, the APlace density objective drives cell spreading. The placement area is divided into grids, each cell has a potential or influence with respect to nearby grids, and the placer seeks to equalize the total cell potential at each grid. For standard-cell placement, the grid usually has a length greater than the average cell width, and the radius of cell's potential, r, is set to be a constant during optimization. However, for mixed-size placement, the size range between large and small objects can be as large as a factor of 10,000 [10], and the radius of influence of a cell's potential will need to change according to the cell's dimension. In particular, a larger block will have potential with respect to more grids.

After investigation of several possibilities, we have chosen to address the potential function for large macros in the following simple way. Suppose a macro block b has width w. The radius or scope of this block's influence is $w/2+r$, i.e., every grid within the distance of $w/2+r$ from the block's center has a non-zero potential from this block. Moreover, the total potential of the block over all grids is equal to the block's area. Therefore, the function $p(d)$ in Equation (2) becomes

$$p(d) = \begin{cases} 1 - a * d^2 & (0 \le d \le w/2 + r/2) \\ b * (d - r)^2 & (w/2 + r/2 \le d \le w/2 + r) \end{cases} \quad (8)$$

where

$$\begin{aligned} a &= 4(w + r^2)/((w + 2r^2)(w + r)^2) \\ b &= 4/(w + 2r^2) \end{aligned} \quad (9)$$

so that the function $p(d)$ is continuous when $d = w/2 + r/2$.

3.3 Legalization

A second key issue is that analytic placement results have cell overlaps that must be legalized. A simplified "Tetris" [17] legalization algorithm is provided in the standard-cell version of APlace; this algorithm also bears strong resemblance to the method proposed in a technical report of Li and Koh [27]. The Tetris legalization is applied after global placement: cells are sorted according to their vertical coordinates, and then for each cell from left to right the current nearest available position is found. This greedy algorithm is very fast, with negligible running time compared to that of global placement.

circuit	cells	nets	macros	pads	$\sum A_m$	A_m^b
ibm01	12752	14111	246	246	67.13	6.37
ibm02	19601	19584	280	259	76.89	11.36
ibm03	23136	27401	290	283	70.75	10.76
ibm04	27507	31970	608	287	59.82	9.16
ibm05	29347	28446	0	1201	0.00	0.00
ibm06	32498	34826	178	166	72.90	13.64
ibm07	45926	48117	507	287	52.56	4.75
ibm08	51309	50513	309	286	67.35	12.11
ibm09	53395	60902	253	285	52.42	5.42
ibm10	69429	75196	786	744	81.37	4.80

Table 1: Benchmark Characteristics.

After investigation of a variety of approaches, we perform legalization in mixed-size placement as follows. Cells are sorted based on a combination of vertical coordinate and width, so that larger blocks may be fixed at a position ahead of nearby small cells. We also scale the cell positions to the left side by a fixed factor (set to 0.90 in all of the discussion and results below) so that (1) cells will not be pushed outside the placement region, and (2) horizontal overlaps among macros can be properly resolved by the legalization.

When an initial global placement has many overlaps among macros, legalization of mixed-size circuits can be extremely challenging. Indeed, a greedy algorithm such as "Tetris" may fail to find a valid position for one or more blocks, or wirelength may be seriously damaged by movement of blocks. Fortunately, the potential function described above allows our mixed-size placer to distribute cells quite evenly in the global placement, with little overlap among larger blocks. Hence, the greedy legalization approach is still an acceptable adjunct even for mixed-size placement: Wirelength increase after the legalization step for our testcases is 6.5% on average (cf. an increase of approximately 5% reported by [21] for pure standard-cell designs).

3.4 Experimental Results

We use ten circuits from the ISPD02 Mixed-Size Placement Benchmarks [18] as our testcases. These circuits are publicly available at the GSRC Bookshelf [6] and are widely used in the literature. The mixed-size designs are relatively large, and contain many macro blocks and standard cells. Total area of macro blocks and standard cells occupies about 80% of the placement area. All macro blocks are assumed to be hard blocks with fixed aspect ratios. Circuit characteristics listed in Table 1 include the number of nets, cells, macro blocks and pads; the total area of macro blocks versus the total area of macro blocks and standard cells, expressed as a percentage ($\sum A_m$); and the area of the biggest block versus the total area of macro blocks and standard cells, expressed as a percentage (A_m^b).

Table 2 summarizes the results for the ten mixed-size circuits. The second and third columns show half-perimeter wirelengths (meters) before and after legalization. The fourth column shows the wirelength increase due to legalization, expressed as a percentage. Average wirelength increase after the legalization step for our testcases is 6.5%. As our placer does not perform detailed placement, we use Feng Shui (v2.4) [25] as the detailed placer. Wirelength (meters) after detailed placement and percentage improvement are shown in the sixth and seventh columns. We see that the average

circuit	our placer				detailed placement		
	WL	WL_l	inc.	CPU	WL_dp	impr.	CPU
ibm01	0.20	0.24	18.5	15	0.23	5.7	1
ibm02	0.51	0.52	0.7	45	0.50	2.5	3
ibm03	0.70	0.74	6.2	56	0.72	3.5	3
ibm04	0.81	0.85	4.8	48	0.83	2.8	4
ibm05	1.01	1.00	-0.5	15	0.98	2.0	5
ibm06	0.65	0.71	9.6	76	0.68	4.4	5
ibm07	1.03	1.09	5.8	98	1.05	3.7	8
ibm08	1.49	1.50	0.6	128	1.46	2.7	8
ibm09	1.25	1.45	15.7	113	1.38	5.2	9
ibm10	2.97	3.07	3.3	206	3.00	2.2	11

Table 2: Results of our placer for ten mixed-size circuits.

circuit	Capo		mPG-MS		Feng Shui		our placer	
	WL	CPU	WL	CPU	WL	CPU	WL	CPU
ibm01	0.31	20	0.30	18	0.24	3	0.23	16
ibm02	0.68	11	0.74	32	0.53	5	0.50	48
ibm03	1.04	59	1.20	32	0.75	6	0.72	59
ibm04	1.01	15	1.05	42	0.80	7	0.83	52
ibm05	1.11	5	1.09	36	1.01	8	0.98	20
ibm06	0.99	18	0.92	45	0.68	10	0.68	81
ibm07	1.53	25	1.37	68	1.17	13	1.05	106
ibm08	1.79	29	1.64	82	1.36	16	1.46	136
ibm09	1.99	29	1.86	84	1.38	15	1.38	122
ibm10	4.55	116	4.36	172	3.75	22	3.00	217

Table 3: Comparison of our results with the Capo flow, mPG-MS and Feng Shui.

wirelength improvement after the detailed placement step is 3.5%. All of our experiments are performed on an Intel Xeon server with dual 2.4GHz CPUs (hyper-threaded) and 4GB memory. Running times of global and detailed placement (minutes) are shown in the fifth and eighth columns, respectively.

Table 3 compares our results with those of recently published works that experiment with the same benchmarks. The results of the three-stage placement-floorplanning-placement flow, which uses the Capo standard-cell placer, were first presented in [1] and further improved in [2]; results of the mPG-MS placer were presented in [10]; and results of Feng Shui (v2.4) are from [25]. Final HPWL values and running times (minutes) for each placer are also shown in Table 3.

According to the results, average wirelength improvement of our placer over the Capo flow is 26.0% (range: 11.5% to 34.0%); average improvement over mPG-MS is 24.7% (range: 9.9% to 40.1%); and average improvement over Feng Shui is 4.0% (range: -7.3% to 20.0%). Running times of the placers cannot be directly compared, since the Capo flow used 2GHz Linux/Pentium 4 workstations, mPG used 750MHz Sun Blade 1000 workstations, and Feng Shui used 2.5GHz Linux/Pentium 4 workstations. With equalized hardware resources, we would expect our placer to be roughly comparable with the Capo flow in runtime, but much slower than Feng Shui. Finally, Figures 2 and 3 show placements for the ibm02 benchmark before and after legalization.

4 Timing-Driven Placement

In this section, we extend the APlace analytic placement framework of [21] to address timing-driven placement.

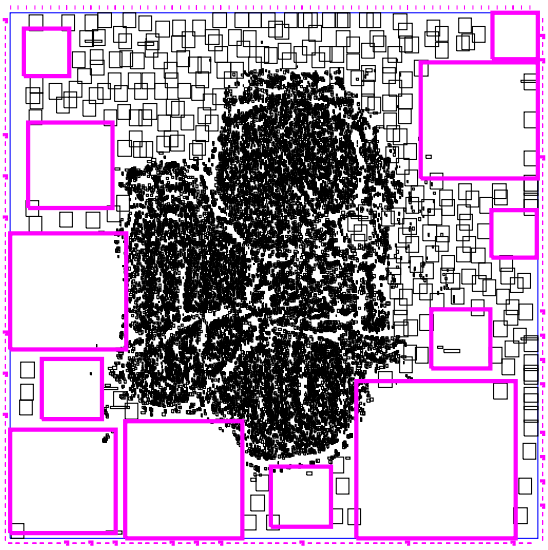

Figure 2: Placement before legalization of our placer for the ibm02 circuit.

4.1 Previous Work

Timing-driven placement has been studied extensively. Existing approaches can be broadly divided into two classes: *path-based* and *net-based*.

A typical path-based approach [16] [20] [34] [35] usually considers all or a subset of paths directly within the problem formulation. The majority of this class of approaches are based on mathematical programming techniques. This class of algorithms usually maintain an accurate timing view during optimization, but its drawback is relatively high complexity due to the exponential number of paths that need to be simultaneously minimized.

For this reason, much of the recent timing-driven work [15] [32] [33] has been net-based. Unlike path-based approaches that handle paths directly, net-based approaches [14] [36] usually transform timing constraints or requirements into either net weight or net length (or delay) constraints, and employ a weighted wirelength minimization engine.

The process of generating net-length constraints or net-delay constraints is called *delay budgeting*. The main idea is to distribute slacks from the end-points of each path to constituent nets along the path, such that a zero-slack solution is obtained [11] [30]. A serious drawback of this class of algorithms is that delay budgeting is usually done in the circuit's structural domain, without consideration of physical placement feasibility. As a result, it may severely over-constrain the placement problem.

Instead of assigning a delay budget to each individual net or edge, net-weighting-based approaches assign weights to nets based on their timing criticality. The basic idea is to put a higher weight for nets that are more timing critical. Net weighting techniques have some favorable properties: relatively low complexity, strong flexibility and easy implementation. As circuit sizes increase and practical timing constraints become increasingly complex, these advantages make the net weighting method more attractive.

There are two principles for assigning net weights. The main principle used in most algorithms is that a timing critical net should receive a heavy weight. For example, VPR [28] uses the following formula to assign weight to an edge e:

$$w(e) = (1 - slack(e)/T)^{\beta} \quad (10)$$

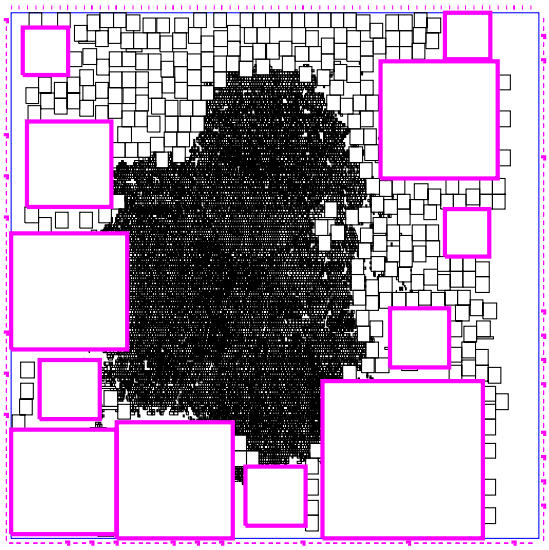

Figure 3: Placement after legalization of our placer for the ibm02 circuit.

where T is the current longest path delay, and β is a constant called the *criticality exponent*.

The other principle is *path sharing* – in general, an edge with many paths passing through should have a heavy weight as well. *Path counting* is a method developed to take path-sharing effects into consideration by computing the number of paths passing through each edge in the circuit. These numbers can then be used as net weights. Another work [26] proposed a solution that distinguishes timing-critical paths from non-critical paths, and scale the impact of all paths by their relative timing criticality. Given a weighting function $D(slack, T)$, the weight assigned to a particular edge e is:

$$w(e) = \sum_{e \in \pi} D(slack(\pi), T) \tag{11}$$

where T is the current longest path delay, and $slack(\pi)$ is the slack of a timing critical path π.

4.2 Slack-Derived Edge Weights

It is natural to apply the net weighting method in APlace to perform timing-driven placement.

Our placer uses the following formula to assign weight $w(e)$ to an edge e:

$$w(e) = 1 + \sum_{e \in \pi} (D(slack(\pi), T) - 1) \tag{12}$$

where

$$D(s,T) = \begin{cases} (1 - s/T)^{\beta} & s \leq 0 \\ 1 & s \geq 0 \end{cases} \tag{13}$$

Here, β is the criticality exponent. $T = (1-u) \cdot \max_{\pi}\{delay(\pi)\}$, $slack(\pi) = T - delay(\pi)$ is the slack of path π and u is the expected improvement of the longest path delay after this timing-driven iteration. The timing-driven process may repeat a few iterations. The weight of a net that is always timing critical is accumulated.

4.3 Timing-Driven Placement Flow

Figure 4 shows the flow of timing-driven process at the final stage of our placer. The near-finished placement of the placer is sent to

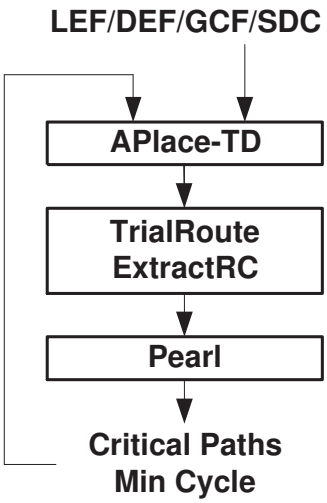

Figure 4: Flow of timing-driven process in APlace-TD.

TrialRoute (SoC Encounter v3.2) to perform a fast global and detailed routing. RC is extracted based on the routing result. Then we use a commercial tool, Pearl (SE v5.4) to do static timing analysis (STA), and the resulting critical path delays are imported into the placer to decide net weights based on timing criticality and path sharing. The weighted wirelength objective is then optimized using the Conjugate Gradient solver, together with the density objective.

4.4 Experimental Results

4.4.1 Impact of Net Weighting Parameters

Experiments are performed to study the parameters of the net weighting formula described in Equations (12) and (13).

Table 4 shows the timing-driven placement runs with varying expected improvements (u's) and criticality exponents (β's) for an industry testcase, indust1. The circuit is in LEF/DEF/GCF/SDC format and has 7077 cells and 8032 nets.

Again, all experiments are performed on an Intel Xeon server with dual 2.4GHz CPUs (hyper-threaded) and 4GB memory. Minimum cycle time (MCT) in nanoseconds is reported by the STA to measure performance of timing-driven placements, together with HPWL (meters) and running times (minutes) of the placements, and routed wirelength (meters) and the number of vias of TrialRoute's results.

The value of expected improvement decides how many timing critical paths are considered for the net weighting. For each value of expected improvement, timing-driven placements are performed with criticality exponents between 1 to 19. The MCT improvement in percentage with timing-driven placement is shown in the last column of Table 4. The minimum cycle time initially decreases with the criticality exponent; since wirelength always increases, the minimum cycle time gradually deteriorates when the criticality exponent is larger.

4.4.2 Comparison with Industry Placers

Results of our placer are compared with two industry placers: QPlace (SE v5.4) and amoebaPlace (SoC Encounter v3.2). We use six industry circuits as our testcases. Two of them, mac1 and mac2, are among the ISPD 2001 Circuit Benchmarks [19] that first appeared in [12]. These circuits are also used in [37] as benchmarks

			Placement			TD-Routing					STA
ckts	**cells**	**nets**	**placer**	**WL**	**CPU**	**over-cap**	**vios**	**WL**	**vias**	**CPU**	**min cycle**
indust1	7077	8032	QPlace	0.59	7	0.19	0	0.74	41517	2	14.67
			QPlace-TD	0.58	21	0.23	0	0.73	41651	1	15.04
			Amoeba	0.59	1	5.80	109	0.86	49191	10	16.04
			Amoeba-TD	0.61	1	8.77	765	0.88	51379	12	14.91
			APlace	**0.51**	**14**	**0.77**	**0**	**0.67**	**43498**	**1**	**14.28**
			APlace-TD	**0.51**	**11**	**0.89**	**0**	**0.68**	**43595**	**2**	**13.83**
indust2	20094	22973	QPlace	1.19	21	2.16	0	2.01	185462	1	48.92
			QPlace-TD	1.29	80	2.80	0	2.31	195440	1	38.87
			Amoeba	1.38	2	2.35	0	2.12	193166	1	59.98
			Amoeba-TD	1.40	5	2.88	0	2.11	194516	2	46.98
			APlace	**1.31**	**58**	**5.09**	**0**	**2.32**	**213298**	**2**	**34.92**
			APlace-TD	**1.31**	**55**	**5.18**	**0**	**2.34**	**214427**	**3**	**33.60**
indust3	40447	42413	QPlace	0.36	20	0.00	0	0.43	278615	4	27.40
			QPlace-TD	0.34	37	0.00	0	0.41	279545	5	27.20
			Amoeba	0.37	3	0.00	0	0.43	302592	2	27.67
			Amoeba-TD	0.36	5	0.00	0	0.42	306896	2	27.31
			APlace	**0.35**	**119**	**0.07**	**0**	**0.43**	**312729**	**3**	**27.65**
			APlace-TD	**0.34**	**112**	**0.05**	**0**	**0.41**	**310976**	**5**	**27.53**
indust4	35272	37543	QPlace	14.56	22	0.01	9	17.32	334235	7	401.50
			QPlace-TD	fail in QPlace-TD							
			Amoeba	15.08	3	0.01	15	16.80	341286	8	401.76
			Amoeba-TD	15.08	6	0.01	21	16.74	341848	8	402.09
			APlace	**12.84**	**65**	**0.01**	**51**	**15.33**	**345492**	**55**	**401.49**
			APlace-TD	**12.80**	**80**	**0.01**	**49**	**15.30**	**344970**	**75**	**401.28**
mac1	5937	6079	QPlace	0.33	3	1.45	0	0.52	56180	1	4.36
			QPlace-TD	0.33	6	1.41	0	0.52	55996	1	4.66
			Amoeba	0.34	1	0.42	0	0.46	56536	1	4.03
			Amoeba-TD	0.36	1	0.35	0	0.47	57827	1	4.46
			APlace	**0.28**	**6**	**0.14**	**0**	**0.40**	**54932**	**1**	**4.13**
			APlace-TD	**0.28**	**9**	**0.10**	**9**	**0.40**	**55236**	**4**	**4.06**
mac2	21491	21766	QPlace	1.30	11	3.39	0	2.43	216182	2	7.46
			QPlace-TD	1.29	22	2.92	0	2.27	211532	2	7.25
			Amoeba	1.38	1	0.34	0	2.23	214249	2	6.15
			Amoeba-TD	1.48	3	0.40	0	2.18	215438	2	6.64
			APlace	**1.15**	**37**	**0.51**	**9**	**2.13**	**220143**	**3**	**6.26**
			APlace-TD	**1.14**	**38**	**0.41**	**16**	**2.11**	**217971**	**6**	**6.18**

Table 5: Timing-driven and non-timing-driven results of our placer for six industry circuits with comparison to QPlace and amoebaPlace.

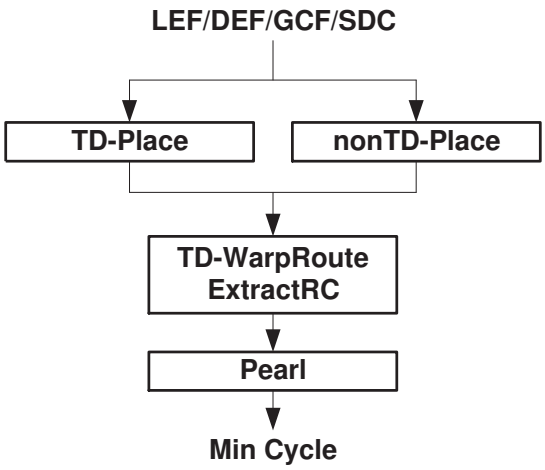

Figure 5: Flow for evaluating quality of timing-driven and non-timing-driven placements.

for timing-driven placement. Only Verilog files are available for these two cases; they are synthesized with a commercial tool, Cadence BuildGates (v5.12). We use a 0.18μm standard-cell library as the LEF file and use the values reported in [37] as the clock cycles for the two circuits. The other four testcases are available in complete LEF/DEF/GCF/SDC format.

Figure 5 shows the flow for evaluating results of timing-driven and non-timing-driven placers in performance quality. Timing-driven and non-timing-driven placements are sent to Cadence WarpRoute (SoC Encounter v3.2) to do timing-driven routing. RC is then extracted and Pearl (SE v5.4) is used to perform static timing analysis (STA). The results are summarized in Table 5. Here, all experiments except those for QPlace are performed on an Intel Xeon server with dual 2.4GHz CPUs (hyper-threaded) and 4GB memory; QPlace is run on a Sun Ultra10 workstation with 400MHz CPU.

In Table 5, the second and third columns show the number of cells and nets of the industry circuits. For each testcase, non-timing-driven and timing-driven placement are performed by each placer. For the indust4 circuit, timing-driven QPlace fails because of incompatible timing constraint file format. Placed wirelengths (meters) and running times (minutes) of each placement are summa-

		Placement		TrialRoute		STA	
u	β	WL	CPU	WL	vias	min cycle	impr.
	0	0.45	11	0.59	34582	14.30	0.00
0.10	3	0.45	12	0.58	34467	13.86	3.08
	5	0.45	12	0.59	34557	13.76	3.78
	7	0.45	12	0.59	34563	13.86	3.08
	9	0.45	12	0.59	34659	13.62	4.76
	11	0.45	12	0.59	34691	13.66	4.48
	13	0.45	12	0.59	34777	13.57	5.10
	15	0.45	12	0.59	34605	13.84	3.22
	17	0.45	12	0.59	34624	13.57	5.10
	19	0.45	11	0.59	34480	13.58	5.03
0.20	1	0.45	12	0.58	34354	13.92	2.66
	3	0.45	13	0.59	34492	13.80	3.50
	5	0.45	13	0.59	34561	13.61	4.83
	7	0.45	13	0.59	34660	13.78	3.64
	9	0.45	12	0.59	34657	13.53	5.38
	11	0.46	12	0.60	34937	13.67	4.41
	13	0.46	12	0.60	35101	13.67	4.41
	15	0.46	12	0.60	35187	13.75	3.85
	17	0.46	12	0.61	35359	13.65	4.55
	19	0.47	12	0.61	35179	13.66	4.48
0.30	1	0.45	13	0.59	34268	13.77	3.71
	3	0.45	13	0.59	34482	13.60	4.90
	5	0.46	13	0.60	34668	13.74	3.92
	7	0.47	13	0.61	35166	13.97	2.31
	9	0.48	12	0.62	35559	13.79	3.57
	11	0.49	12	0.63	35772	13.81	3.43
	13	0.50	13	0.64	36093	13.95	2.45
	15	0.52	12	0.68	37232	13.99	2.17
	17	0.54	12	0.69	37632	13.88	2.94
	19	0.61	13	0.80	39339	14.46	-1.12

Table 4: Timing-driven results with varying expected improvements (u's) and criticality exponents (β's) for the indust1 circuit.

rized in the fifth and sixth columns. APlace-TD usually has a better placed wirelength than industry placers; but it is slower.

Timing-driven routing is performed with WarpRoute; overcapacity gcells in percentage, the number of violations, routed wirelength (meters), the number of vias and running times (minutes) of WarpRoute's results are shown in the seventh through eleventh columns of Table 5. Most of the placement results of APlace-TD can be successively routed with good wirelength; finished routings with a small number of violations can be manually fixed. According to the results, average (routed) wirelength improvement of APlace-TD over QPlace is 7.2% (range: -1.2% to 7.1%); and average improvement over amoebaPlace is 6.5% (range: -11.1% to 23.2%). Compared to non-timing-driven APlace, APlace-TD has a slightly better routed wirelength (0.7% on average).

In the last column, minimum cycle time (nanoseconds) is reported from the STA tool as the performance measure of timing-driven or non-timing-driven placements.[2] Our placer usually has a better minimum cycle time. Average improvement in minimum cycle time of APlace-TD over QPlace is 9.6% (range: -1.2% to 14.8%); and average improvement over amoebaPlace is 8.5% (range: -0.8% to 28.5%). Compared to non-timing-driven APlace, APlace-TD improves the minimum cycle time by 2% on average (range: 0.1% to 3.8%). The MCT improvements are especially negligible for the indust3 and indust4 circuits; as noted in the footnote above, minimum cycle times of these two circuits are less sensitive with different net weights.

[2] For some testcases, especially mac1 and mac2, timing-driven placements from the industry placement can have worse minimum cycle times than those from non-timing-driven placement. This may be due to improper timing constraints in the testcases - e.g., for "old" designs. We also note that the indust3 and indust4 testcases do not show any significant sensitivity of MCT to the placement.

5 Conclusion and Future Work

Motivated by the needs of leading-edge chip implementation methodologies, we have extended the APlace wirelength-driven standard-cell analytic placement framework of [21] into a new placer, APlace-TDMS, which addresses both timing-driven and mixed-size placement. For mixed-size placement, APlace-TDMS achieves half-perimeter wirelength outperforming that of mPG-MS, Feng Shui and the Capo flow respectively by 24.7%, 4.0% and 26.0% on average. For timing-driven placement, APlace-TDMS achieves minimum cycle time that outperforms that of QPlace and amoebaPlace respectively by 9.6% and 8.5%, while simultaneously achieving average improvements of 7.2% and 6.5% in routed wirelength, respectively. An obvious weakness in our validation of currently available testcases is that we have been able to test APlace-TDMS on several industry timing-driven testcases, and on several public-domain mixed-size testcases - but not on any timing-driven mixed-size testcases. Indeed, to our knowledge there are no public benchmarks available for this purpose. To remedy this, we have recently obtained and are currently evaluating our placer on industry testcases that combine the challenges of timing constraints, large instance sizes, and embedded blocks (both fixed and unfixed). Beyond this benchmarking effort, our ongoing research directions include: (1) extension of the placer to power or IR drop directed placement; (2) extension to 3D placement; (3) extension to thermal-directed placement; and (4) devising a unified analytic placement approach that can simultaneously address congestion, timing, power, and wirelength at a level beyond the existing state of the art.

References

[1] S. N. Adya and I. L. Markov, "Consistent Placement of Macroblock Using Floorplanning and Standard-Cell Placement", *Proc. Int. Symp. Physical Design*, 2002, pp. 12-17.

[2] S. N. Adya, I. L. Markov and P. G. Villarrubia, "On Whitespace in Mixed-Size Placement and Physical Sysnthesis", *Proc. IEEE/ACM Int. Conf. Computer Aided Design*, 2003, pp. 311-318.

[3] C. J. Alpert, T. Chan, A. B. Kahng, I. Markov and P. Mulet, "Faster Minimization of Linear Wirelength for Global Placement", *IEEE Trans. Computer Aided Design* 17(1) (1998), pp. 3-13.

[4] C. J. Alpert, G.-J. Nam and P. G. Villarrubia, "Free Space Management for Cut-based Placement", *Proc. IEEE/ACM Int. Conf. Computer Aided Design*, Nov. 2002, pp. 746-751.

[5] R. Baldick, A. B. Kahng, A. Kennings and I. L. Markov, "Function Smoothing with Applications to VLSI Layout", *Proc. Asia and South Pacific Design Automation Conf.*, Jan. 1999, pp. 225-228.

[6] Web site of VLSI CAD Bookshelf. `http://www.gigascale.org/bookshelf/`.

[7] A. E. Caldwell, A. B. Kahng, and I. L. Markov, "Can Recursive Bisection Alone Produce Routable Placements?", *Proc. ACM/IEEE Design Automation Conf.*, 2000, pp. 477-482.

[8] A. E. Caldwell, A. B. Kahng, A. A. Kennings and I. L. Markov, "Hypergraph Partitioning for VLSI CAD: Methodology for Reporting, and New Results", *Proc. ACM/IEEE Design Automation Conf.*, June 1999, pp. 349-354.

[9] Web site of UCLA/UMICH Physical Design Tools. `http://vlsicad.eecs.umich.edu/BK/PDtools/`.

[10] C. C. Chang, J. Cong and X. Yuan, "Multi-Level Placement for Large-Scale Mixed-Size IC Designs", *Proc. Asia South Pacific Design Automation Conf.*, 2003, pp. 325-330.

[11] C. Chen, X. Yang and M. Sarrafzadeh, "Potential Slack: An Effective Metric of Combinational Circuit Performance", *Proc. IEEE/ACM Int. Conf. Computer Aided Design*, 2000, pp. 198-201.

[12] Y.-C. Chou and Y.-L. Lin, "A Performance-Driven Standard Cell Placer Based on a Modified Force-Directed Algorithm", *Proc. Int. Symp. Physical Design*, 2001, pp. 24-29.

[13] Web site of Dragon. `http://er.cs.ucla.edu/Dragon/`.

[14] H. Eisenmann and F. M. Johannes, "Generic Global Placement and Floorplanning", *Proc. ACM/IEEE Design Automation Conf.*, 1998, pp. 269-274.

[15] B. Halpin, C.-Y. R. Chen and N. Sehgal, "Timing Driven Placement using Physical Net Constraints", *Proc. ACM/IEEE Design Automation Conf.*, 2001, pp. 780-783.

[16] T. Hamada, C. K. Cheng, and P. M. Chau, "Prime: A Timing-Driven Placement Tool Using a Piecewise Linear Resistive Network Approach", *Proc. ACM/IEEE Design Automation Conf.*, 1993, pp. 531-536.

[17] D. Hill, "Method and System for High Speed Detailed Placement of Cells within an Integrated Circuit Design", *US Patent* 6370673, April 2002.

[18] Web site of ISPD02 Mixed-Size Placement Benchmarks. `http://vlsicad.eecs.umich.edu/BK/ISPD02bench/`.

[19] Web site of ISPD 2001 Circuit Benchmarks. `http://nthucad.cs.nthu.edu.tw/~ycchou/benchmark/placement.htm`.

[20] M. Jackson and E. S. Kuh, "Performance-Driven Placement of Cell Based IC's", *Proc. ACM/IEEE Design Automation Conf.*, 1989, pp. 370-375.

[21] A. B. Kahng and Q. Wang, "Implementation and Extensibility of an Analytic Placer", *Proc. Int. Symp. Physical Design*, 2004, pp. 18-25.

[22] A. B. Kahng and X. Xu, "Accurate Pseudo-Constructive Wirelength and Congestion Estimation", *Proc. ACM Int. Workshop on System-Level Interconnect Prediction*, 2003, pp. 61-68.

[23] A. A. Kennings and I. L. Markov, "Smoothening Max-terms and Analytical Minimization of Half-Perimeter Wirelength", *VLSI Design* 14(3) (2002), pp. 229-237.

[24] A. A. Kennings and I. L. Markov, "Analytical Minimization of Half-Perimeter Wirelength", *Proc. Asia and South Pacific Design Automation Conf.*, Jan. 2000, pp. 179-184.

[25] A. Khatkhate, C. Li, A. R. Agnihotri, M. C. Yildiz, S. Ono, C.-K. Koh and P. H. Madden, "Recursive Bisection Based Mixed Block Placement", *Proc. Int. Symp. Physical Design*, 2004, pp. 84-89.

[26] T. Kong, "A Novel Net Weighting Algorithm for Timing-Driven Placement", *Proc. IEEE/ACM Int. Conf. Computer Aided Design*, 2002, pp. 10-14.

[27] C. Li and C.-K. Koh, "On Improving Recursive Bipartitioning-Based Placement", *Technical Report* TR-ECE-03-14, Purdue Univ., 2003.

[28] A. Marquardt, V. Betz and J. Rose, "Timing-Driven Placement for FPGAs", *Proc. ACM Symp. FPGAs*, 2000, pp. 203-213.

[29] K. G. Murty, *Linear Complementarity, Linear and Nonlinear Programming*, Internet Edition, Chapter 10, pp. 389-460. `http://ioe.engin.umich.edu/books/murty/linear_complementarity_webbook/`.

[30] R. Nair, C. L. Berman, P. Hauge and E. J. Yoffa, "Generation of Performance Constraints for Layout", *IEEE Trans. Computer Aided Design of Integrated Circuits and Systems* 8(8) (1989), pp. 860-874.

[31] W. Naylor et al., "Non-Linear Optimization System and Method for Wire Length and Delay Optimization for an Automatic Electric Circuit Placer", *US Patent* 6301693, Oct. 2001.

[32] S.-L. Ou and M. Pedram, "Timing-Driven Placement Based on Partitioning with Dynamic Cut-Net Control", *Proc. ACM/IEEE Design Automation Conf.*, 2000, pp. 472-476.

[33] K. Rajagopal, T. Shaked, Y. Parasuram, T. Cao, A. Chowdhary and B. Halpin, "Timing Driven Force Directed Placement with Physical Net Constraints", *Proc. Int. Symp. Physical Design*, 2003, pp. 60-66.

[34] A. Srinivasan, K. Chaudhary and E. S. Kuh, "RITUAL: A Performance Driven Placement for Small-Cell ICs", *Proc. IEEE/ACM Int. Conf. Computer Aided Design*, 1991, pp. 48-51.

[35] W. Swartz and C. Sechen, "Timing Driven Placement for Large Standard Cell Circuits", *Proc. ACM/IEEE Design Automation Conf.*, 1995, pp. 211-215.

[36] R. S. Tsay and J. Koehl, "An Analytic Net Weighting Approach for Performance Optimization in Circuit Placement", *Proc. ACM/IEEE Design Automation Conf.*, 1991, pp. 620-625.

[37] X. Yang, B.-K. Choi and M. Sarrafzadeh, "Timing-Driven Placement Using Design Hierarchy Guided Constraint", *Proc. IEEE/ACM Int. Conf. Computer Aided Design*, 2002, pp. 177-180.