Structure-Aware Placement Techniques for Designs With Datapaths

Samuel I. Ward, *Student Member, IEEE,* Myung-Chul Kim, *Member, IEEE,* Natarajan Viswanathan, *Member, IEEE,* Zhuo Li, *Senior Member, IEEE,* Charles J. Alpert, *Fellow, IEEE,* Earl E. Swartzlander, Jr., *Life Fellow, IEEE,* and David Z. Pan, *Senior Member, IEEE*

***Abstract*—As technology scales and frequencies increase, a new hybrid design style emerges, wherein designs contain a mixture of random logic and datapath standard-cell components. This paper demonstrates that conventional half-perimeter wirelength driven placers underperform in terms of regularity and Steiner wirelength (StWL) for such hybrid designs. In addition, the quality gap between manual and automatic placement is more pronounced as the designs become more datapath oriented. To effectively handle hybrid designs, this paper proposes a new unified placement flow that simultaneously places random logic and datapath cells. This flow is built on the top of a leading academic force-directed placer and significantly improves the quality of datapath placement while leveraging the speed and flexibility of existing random-logic placement algorithms. It consists of a suite of novel global and detailed placement techniques, collectively called structure-aware placement techniques (SAPT). These techniques effectively integrate alignment constraints into placement, thereby overcoming the deficiencies of existing random-logic placers when handling designs with embedded datapaths. Compared to other state-of-the-art placers, SAPT improves total StWL by more than 28% and total routing overflow by over six times on the ISPD 2011 datapath benchmark suite. In addition, it improves total StWL by 5.8% on industrial hybrid designs.**

***Index Terms*—Algorithms, datapath, layout, optimization, physical design, placement.**

I. Introduction

As APPLICATION-SPECIFIC integrated circuit frequencies exceed 1 GHz and shrinking schedules drive increased automation for microprocessor designs, the boundary between manually designed datapath-logic and random-logic macros is blurring. A new hybrid design style is emerging, wherein designs contain both random logic and datapath logic. The datapath logic generally refers to circuit structures containing highly parallel bit operations [1] (often called the bit stack), and careful design is important for high frequency designs. Prior work [2] has shown that handling the datapath-logic placement independent of the random logic, overly constrains the random-logic placement, degrading overall congestion and wirelength. A single placement flow handling both datapath and random logic is extremely valuable, improving design time, solution quality, and saving development and maintenance costs. However, [3], [4] demonstrate that most state-of-the-art placers are incapable of handling designs with regular structure. This paper shows that with design guidance, existing half-perimeter wirelength (HPWL)-driven placers can better handle designs with embedded datapath logic.

This paper presents a novel structure-aware placement flow for hybrid designs via a set of effective placement techniques amenable to incorporation within existing random-logic placers. The flow leverages the speed and flexibility of state-of-the-art HPWL-driven placers, while imposing alignment constraints[1] to achieve better regularity and Steiner wirelength (StWL). The key contributions of this paper are as follows.

1) A study of the issues with current academic placers: the inadequacies and specifically the lack of fidelity of the HPWL model versus the StWL model when evaluating and placing datapath logic.
2) A key insight to bit-stack alignment: alignment of the bit-stack guides indirect StWL optimization, and significantly improves total StWL and routing congestion.
3) A novel placement flow: structure-aware placement techniques (SAPT) that can be incorporated within existing HPWL-driven placers to enable better alignment of the embedded datapaths during both global and detailed placement.

Section II outlines the problem faced by current random-logic placers when placing datapath logic. Section III presents the preliminaries and placement definitions. Section IV provides an overview of the structure-aware placement flow with general descriptions of each technique. The structure-aware global placement techniques are described in Section VI and structure-aware detailed placement techniques are described in Section VII. Section VIII presents experimental results, followed by conclusions and future work in Section IX. This is an extension of the preliminary work presented in [6].

II. Motivation and Background

A common assumption among integrated circuit (IC) designers is that circuits with high regularity such as datapath logic require manual placement. Perpetuating this assumption

Manuscript received June 15, 2012; revised August 31, 2012 and October 19, 2012; accepted November 12, 2012. Date of current version January 18, 2013. This paper was recommended by Associate Editor C.-K. Koh.

S. I. Ward, E. E. Swartzlander, Jr., and D. Z. Pan are with the University of Texas, Austin, TX 78712 USA (e-mail: wardsi@utexas.edu; eswartzla@aol.com; dpan@ece.utexas.edu).

M.-C. Kim and N. Viswanathan are with IBM Corporation, Austin, TX 78758 USA (e-mail: myungk@us.ibm.com; nviswan@us.ibm.com).

Z. Li and C. J. Alpert are with the IBM Austin Research Laboratory, Austin, TX 78758 USA (e-mail: lizhuo@us.ibm.com; alpert@us.ibm.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2012.2233862

[1] Alignment constraint was also discussed in [5] as an example of geometric constraint handling, but no circuit structures were considered.

are two key factors that limited adoption of past automation attempts. First, prior approaches separate control logic placement from datapath-logic placement. Second, a prevailing evaluation metric for random-logic placement, HPWL is inadequate for structured circuit styles. This section addresses each of these factors, by first establishing the need for a unified placement framework and then highlighting the inadequacy of the HPWL metric for regular structures. It then demonstrates that cell alignment during placement implicitly optimizes StWL, producing significant wirelength improvements for datapath style circuits.

A. Need for a Unified Placement Framework

Automatic placement of structured circuits has been performed by dedicated datapath placers such as [1], [7], [8], which generate highly compact, area efficient placements. After layout generation, these methods construct a larger macro block or small individual bit-slice macro blocks, followed by the main random-logic mixed-size placer. More recently, promising results were presented in [9] where an innovative row-based placement of the datapath is proposed instead of the traditional bit-stack alignment. A nonlinear optimization for HPWL minimization with a sigmoid-based density model for density control in datapath circuits is proposed. Once datapath placement completes, the cells become a movable macro. Generalizing this approach, results show that it is possible to separately place the datapath cells and then apply mixed-size placement techniques to generate significantly smaller HPWL than prior placers.

However, these techniques suffer from some key drawbacks. First, even though a datapath placer may minimize the local wirelength through cell ordering [10], or optimizing specific bit stacks [11], global interconnect optimization with the embedded datapath is not taken into account simultaneously during placement. Second, by making the datapath a macro, in the general case, the datapath macro layout must occur first and it is very difficult to select and optimize the correct macro aspect ratio. Third, in industrial hybrid designs, the datapath is not always tightly packed but many cases still require alignment. Fourth, packing may force other more critical random logic out of a specific area, reducing overall result quality. Thus, though very promising, significant future work is ahead for evaluating these techniques within a full-industrial physical design flow.

B. StWL and HPWL Comparisons for Datapath Circuits

Datapath circuits are typically driven by one or more high-fanout nets. Traditional HPWL-driven placers naturally compact the placement of high-fanout nets to minimize total wirelength. However, known optimal layouts for many regular datapath structures are drastically different [18], often not corresponding to a minimum-HPWL placement solution. To illustrate this point, Table I compares a few state-of-the-art academic placers using both, total HPWL and total StWL on the modified ISPD 2011 Datapath Benchmark Suite [19].[2] All StWL measurements were performed using coalesCgrip [20], and all reported numbers are total wirelength results for each design. The HPWL column in Table I is sorted from the smallest to the largest for each benchmark. In addition, the table reports the wirelength ratio normalized to the manually placed solution. Careful examination of this table yields the following surprising results.

1) While HPWL from the automated placement solutions for both benchmarks are very close to the manually placed solution, the StWL results degrade significantly, with the best automated solution at 1.82× in StWL for benchmark A and 2.27× for benchmark B compared to the manual solution.
2) Fidelity of the HPWL metric appears low for datapath logic. As shown in Table I, the HPWL column is sorted by increasing value and it is generally expected that StWL would maintain the same order, but in fact that does not happen. Additionally, for both benchmarks, the placer with the best HPWL (Capo) is not the placer with the best StWL (SimPL).

As shown in Section VIII-D, the significant improvement in StWL also corresponds to vastly improved congestion metrics. There has been prior work in directly optimizing StWL [21] with the Rooster placer. As reported in [21], StWL has much better correlation to the routed wirelength (rWL) as compared to HPWL. However, as results show in Table VII for Rooster, optimizing StWL alone does not effectively address the alignment requirements of datapath circuits. Additionally, HPWL is easy to compute, and is a reasonable first-order estimate of timing and power on the vast number of random-logic designs. This makes it a popular objective to optimize during placement. Therefore, instead of completely changing the placement objective, this paper presents techniques to improve the placement quality of datapath logic under the hood of existing HPWL-driven placement frameworks.

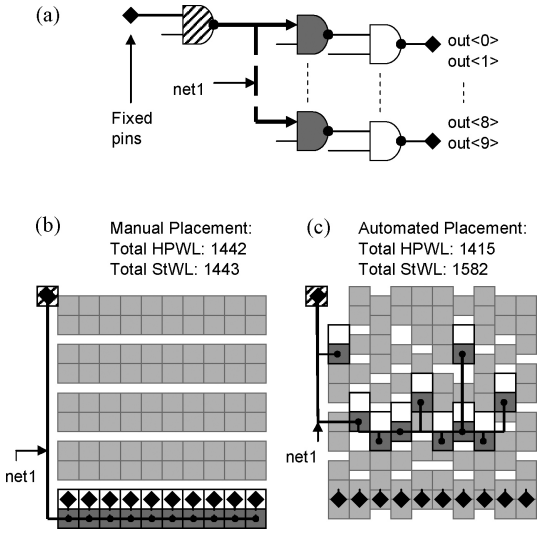


Fig. 1. Example circuit where StWL of the manually placed design is better than that of the automated placement, but HPWL of the automated placement solution is better than that of the manual placement. Net1 has fanout of 10.

C. Implicit StWL Optimization Through Bit-Stack Alignment

In Fig. 1(a), a partial logic netlist with one NAND gate, shown as hashed, drives net *net*1 with a fanout of 10. All the input and output pins are fixed objects placed on top of the gate. Fig. 1(b) shows a manually placed solution for this partial

[2]The MISPD 2011 Datapath Benchmark Suite was modified to contain unfixed latch rows compared to the original fixed latch placement reported in ISPD 2011. Benchmarks can be downloaded at http://www.cerc.utexas.edu/utda/download/DP/ [3].

TABLE I
LEGALIZED HPWL AND STWL COMPARISON ON THE ISPD 2011 DATAPATH BENCHMARK SUITE [12] BETWEEN MANUALLY PLACED AND AUTOMATED PLACEMENT SOLUTIONS

	ISPD Datapath Benchmark A					ISPD Datapath Benchmark B			
	Total HPWL		Total StWL			Total HPWL		Total StWL	
Manually placed	11 000 365	1.00	11 066 683	1.00	Manually Placed	8 642 097	1.00	9 823 680	1.00
CAPO v10.2 [12]	11 535 525	**1.05**	21 516 128	1.94	CAPO v10.2	10 338 805	**1.20**	23 881 606	2.43
SimPL [13]	11 837 307	1.08	20 180 311	**1.82**	NTUPlace3 v7.10.19	10 433 894	1.21	26 110 039	2.66
mPL6 v6 [14]	12 919 955	1.17	23 950 663	2.16	SimPL	10 631 304	1.23	22 319 594	**2.27**
NTUPlace3 v7.10.19 [15]	13 447 753	1.22	24 673 151	2.23	Dragon v3.01	12 229 019	1.42	28 577 316	2.91
FastPlace v3.0 [16]	15 672 727	1.42	27 115 750	2.45	FastPlace v3.0	14 537 026	1.68	36 642 434	3.73
Dragon v3.01 [17]	16 424 739	1.49	26 182 449	2.37	mPL6 v6	16 263 018	1.88	28 846 387	2.94

Placement results are sorted by increasing HPWL value. Note that: 1) best HPWL solution does not indicate the best StWL solution, and 2) bold numbers are the best automated placement wirelength.

circuit and Fig. 1(c) shows a solution from an existing placer. The dark-shaded cells match the same dark-shaded NAND gates in Fig. 1(a). The light-shaded gray cells represent other logic placed within the design.

For both solutions, the total HPWL and StWL numbers are shown in Fig. 1. As indicated in Section II-B, even though the HPWL of the manual solution (1442) is greater than the HPWL of the automated placement (1415), the StWL shows the reverse trend. While it is impractical to list HPWL and StWL of every single net, clearly for net $net1$, the StWL in Fig. 1(b) is better than the StWL in Fig. 1(c). This is due to the better alignment of the structured cells in one horizontal row, which produces much better StWL. Also the solution of Fig. 1(c) shows the existing placer compacting the placement of the net in both the x- and y-directions to lower HPWL, but degrading StWL. This example shows that if a HPWL-driven placer can obtain better alignment for regular structures, it can implicitly have better StWL, without having to optimize for it directly.

Motivated by the above examples, new techniques are developed to guide an existing HPWL-driven random-logic placer to generate a placement similar to Fig. 1(b), with better StWL than the one in Fig. 1(c). Additionally, by providing alignment constraints to small portions of the datapath, it is observed that during the iterative placement process, other surrounding cells become aligned as well. This can be observed visually in the placement results in Fig. 11 where only some of the cells have been manually defined.

The alignment constraints presented in this paper provide hints to placers, directing them toward more globally optimized solution. As the results will show, with relatively few manually defined bit stacks, our framework significantly reduces overall wirelength and congestion.

III. PRELIMINARIES

Given a netlist $N = (V, E)$ with nodes V and nets E, placement obtains locations (x_i, y_i) for all the movable nodes, such that the area of nodes within any rectangular region does not exceed the area of cell sites in that region.

With $\vec{x}, \vec{y} = \{x_i, y_i\}$, the HPWL is defined as

$$HPWL(\vec{x}, \vec{y}) = HPWL(\vec{x}) + HPWL(\vec{y}) \quad (1)$$

$$HPWL(\vec{x}) = \sum_{e \in E} [MAX_{i \in e} x_i - MIN_{i \in e} x_i]. \quad (2)$$

Typically, force-directed placers optimize a quadratic approximation of the HPWL

$$\Phi_G(\vec{x}, \vec{y}) = \sum_{i,j} w_{i,j}[(x_i - x_j)^2 + (y_i - y_j)^2]. \quad (3)$$

From (3), (x_i, y_i) represents the coordinates of cell i, and $w_{i,j}$ represents the net weight of the connection between cells i and j.

In this paper, a force-directed global placer in the spirit of SimPL [13], where $w_{i,j}$ is calculated by the Bound2Bound net model [22], is used along with a detailed placer derived from FastPlace-DP [23]. SimPL is a flat force-directed global placer. It maintains a lower bound and an upper bound placement and progressively narrows the displacement between the two to yield a final placement solution. The upper bound placement is generated by applying lookahead legalization (LAL), which is based on top-down geometric partitioning and nonlinear scaling. The coordinates obtained from the upper bound placement are used as fixed points, which are connected to their corresponding cells via pseudo nets to provide spreading forces. The lower bound placement is then generated by minimizing the quadratic objective in (3).

A. Alignment Groups

Definition 1: An alignment group $g_k \in G$ where $1 \leq k \leq |G|$, is an unordered subset of cells from V. An alignment direction $\vec{d}_k$ is a preferred placement direction of g_k, where $0 \leq \vec{d}_k \leq 90$, with 0 representing horizontal and 90 representing the vertical direction.

Essentially, an alignment group is a set of cells that need to be aligned in a certain direction (e.g., horizontally or vertically) during placement to optimize the datapath. In this paper, it is assumed that the set of alignment groups G, and their alignment directions are given. Additionally, the collection of g_k with the same d_k value is pairwise disjoint.

Generally, g_k may correspond to bit stacks in the datapath, but can be other elements such as cells connected to a single high-fanout net that improves through alignment, buffers that need careful placement to facilitate routing of large buses, or pipelining latches. For alignment directions, in this paper, only horizontal and vertical directions are considered, which means $\vec{d}_k \in \{0, 90\}$. The above assumptions that alignment groups and directions can be given are valid and practical. One may use datapath extractors such as [24]–[26] to extract the alignment groups based on circuit properties. Alternatively, this information may come directly from logical descriptions of the netlist, or could be provided by designers. As an example, if designers are trying to structure the placement of latches, it is trivial for them to provide sets of latch names and their preferred placement directions.

B. Pseudo Nets

Fixed-point generation followed by pseudo net insertion is a common method to apply spreading forces during iterative force-directed placement.

Definition 2: A pseudo net $c(f, i)$ is a weighted two-pin connection between a fixed-point f and a cell i in the circuit netlist. The pseudo net has a weight equal to $\epsilon \cdot w_{i,j}$, defined in [13], and does not exist in the circuit netlist.

Please note that these nets are added for every movable cell in the design. In addition, existing pseudo nets are discarded at the end of the current iteration, and a new set is added to enforce spreading during the subsequent placement iteration. The pseudo net-weighting technique with varying ϵ is described in [13], [27], and controls the rate of overlap removal during global placement. During early iterations, greater significance is given to interconnect minimization, while the relative cell ordering stabilizes. This is accomplished by starting with a small ϵ value and gradually increasing it through each iteration. This scheme provides flexibility to the placer during the early stages, while tightening the constraints to resolve overlap toward the end of global placement. The theoretical justification and extensions of the SimPL framework is provided in [27].

C. Alignment Nets

To achieve better datapath alignment, one approach is direct manipulation of existing nets between the datapath cells. However, this approach interferes with other placement directives. Specifically, direct weighting manipulation of current nets disrupts timing-driven/power-driven placement and net weighting for those cells. Hence, a new category of nets is introduced, referred to as alignment nets, and defined as follows.

Definition 3: An alignment net s_k, where $1 \leq k \leq |G|$, is a weighted multipin connection among all cells in an alignment group g_k. For placement, this multipin alignment net is decomposed into 2-pin nets, where netweights are found by the Bound2Bound net model [22].

Alignment nets are created at the beginning of placement and remain persistent during the entire global and detailed placement stages. A skewed-netweight scheduling helps these nets align the cells within the corresponding alignment group g_k inside the layout region. By applying alignment constraints to new nets s_k, prior directives relying on net weighting continue to function as before.

IV. Unified Placement Flow With Alignment Constraints

Fig. 2 presents a unified structure-aware placement flow, referred to as SAPT, which simultaneously places both datapath and random logic. The shaded boxes represent the steps in a conventional force-directed placement flow and the white boxes represent enhancements to better handle datapath circuits. The key algorithmic components of SAPT from the flow diagram are as follows.

1) *Alignment Net Insertion:* Alignment nets are inserted to manipulate placements of a specific set of cells during global placement (Section III-C).
2) *Alignment Group Order Extraction:* On highly structured circuits, it is possible to extract relative positions of alignment groups to better optimize placement. A technique is presented to automatically extract the relative order between alignment groups prior to placement (Section V).

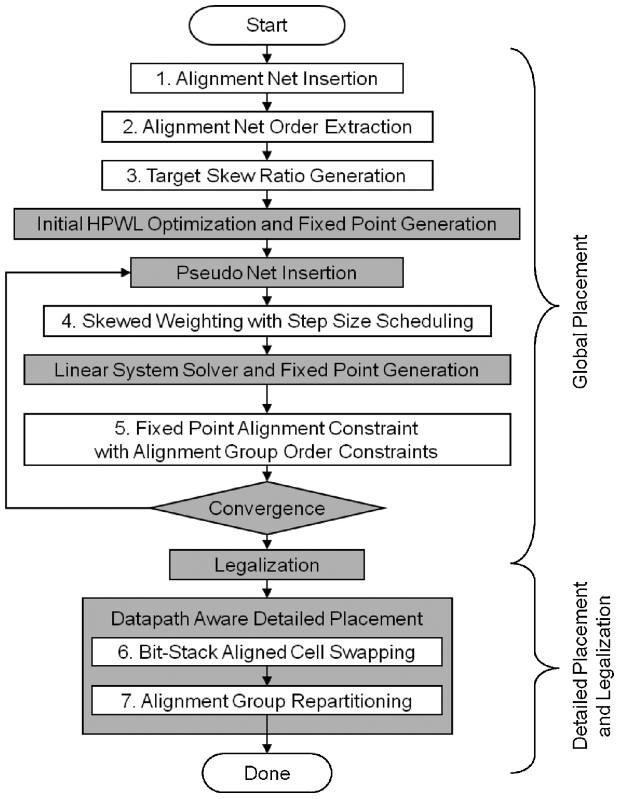


Fig. 2. Proposed unified datapath-aware placement flow. The baseline components are shown in shaded boxes and the newly added datapath-aware components are shown in transparent boxes.

3) *Target Skew Ratio Generation:* During placement, the alignment net weights are modulated to optimize the datapath placement. A method to automatically generate the maximum netweights in the horizontal and vertical directions is developed to achieve a desired spread or span for an alignment net over the layout region (Section VI-B).
4) *Skewed Weighting With Step Size Scheduling:* This describes the skewed net-weighting process applied to each alignment net s_k to gradually improve alignment along the datapath. The weight in a particular direction gradually increases during placement iterations until it reaches the target skew for that direction (Section VI-A).
5) *Fixed-Point Alignment Constraint with Alignment Group Order Constraints:* Force-directed global placement frameworks reduce cell overlap by using fixed-points and pseudo nets. When a placer is structure-unaware, this process leads to misalignment for datapath logic. This section describes a process to modify the fixed-point locations to improve structured placements. Additionally, fixed-point alignment constraints generate a relative order among alignment groups that is not always optimal. This step modifies the fixed-point positions and maintains the extracted relative order among alignment groups (from Section V) during global placement (Section VI-C).
6) *Bit-Stack Aligned Cell Swapping:* Generating alignment during global placement does not guarantee that cells within the alignment group will remain aligned during detailed placement. The global cell swapping step of de-

Algorithm 1 Alignment Group Order Extraction

Inputs : A netlist N, alignment groups g_k
Output : A totally ordered set of alignment groups for each $\vec{d}_k \in \{0, 90\}$
Stage 1: Extraction of Intrasupergroup Orders
1: Identify logically independent supergroups by graph traversal.
2: Find a total order of g_k within a supergroup based on fanin/out relations.
Stage 2: Extraction of an Intersupergroup Order
3: Identify PO pins in the supergroups with the same smallest logic distance.
4: Determine a total order of supergroups by spatial locations of the PO pins.
Stage 3: Derivation of an Alignment Group Order
5: Derive a total order of g_k from the intra- and inter-supergroup orders.

tailed placement is enhanced to maintain the alignment generated during global placement (Section VII-A).

7) *Alignment Group Repartitioning:* With datapaths, traditional HPWL metrics can at times fail to detect alignment improvements. This technique minimizes internal net cut values, potentially improving both HPWL and StWL metrics for all nodes in g_k along s_k (Section VII-B).

At each step, the modifications apply only to the defined alignment groups g_k (Section III-A) leaving all other random-logic cells to be placed as they would before. Though in this paper SimPL and FastPlace-DP are used as an example, the techniques can be adapted to other force-directed placement algorithms. Section V describes preprocessing steps and then a detailed description of the global and detailed placement techniques are presented in Sections VI and VII, respectively.

V. Alignment Group Order Extraction

In addition to the misalignment of alignment groups, their relative orders within the placement region can contribute to the suboptimality of automated datapath placement. To this end, techniques are developed for extracting good relative orders among alignment groups and maintaining them during placement iterations to better optimize the resulting placements. This section describes the preprocessing step of alignment group order extraction, required for supplying alignment constraints to a set of cells within the netlist. Relative orders extracted at the beginning of global placement are based on a given netlist and fixed pin locations, which are persistent regardless of intermediate placements. First, alignment group order is defined for each $\vec{d}_k \in \{0, 90\}$.

Definition 4: An alignment group order is a binary relation defined on a subset of G that shares the same $\vec{d}_k$. It is a strict partial order among a pair of g_i and g_j for $1 \leq i, j \leq |G|$, which directly corresponds to directed acyclic graphs.

For each $\vec{d}_k$, extraction of the order among alignment groups is performed in three stages: 1) extraction of relative orders among alignment groups based on graph traversal [e.g., depth-first search (DFS)]; 2) extraction of relative order among logically independent supergroups based on primary input (PI)/primary output (PO) pin locations; and 3) derivation of a total order of alignment groups (Algorithm 1).

In the first stage of extraction, sets of logically related alignment groups per $\vec{d}_k$ are identified and a unique order within the sets is determined. According to Definition 4, the process begins with constructing a directed acyclic graph based on the netlist and traversing through each defined alignment group. Starting from a cell in an alignment group, DFS identifies a set of logically related alignment groups with an order (via fanin/fanout relations), denoted by supergroup.

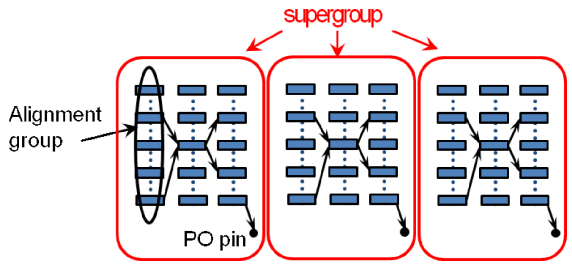


Fig. 3. Examples of alignment groups ($\vec{d}_k = 90$) and supergroups. Alignment nets connect cells in alignment groups, depicted by dotted lines. Solid arrows represent fanin/fanout connections, used to define partial alignment group orders. The supergroup order obtained by referring to the associated PO pin locations then finds a total alignment group order.

Definition 5: A supergroup is a totally ordered set of logically related alignment groups with the same $\vec{d}_k$. Supergroups are pairwise disjoint.

This way, multiple logically independent supergroups can be extracted where a unique alignment group order within each supergroup is defined. Since the resulting order within a supergroup is logical, alignment groups can be either placed from left to right or from right to left for g_k with $\vec{d}_k = 90$. Without loss of generality, they are placed left to right when $\vec{d}_k = 90$ and bottom to top when $\vec{d}_k = 0$.

Unlike other random logic-oriented designs, in datapath designs, there can exist multiple supergroups that are logically independent while having the symmetry in their structure. For example, as seen from the Structured Placement Benchmark A in Fig. 4 from [3], the first m bit MUX select signals form a supergroup, and such supergroups are repeated many times at different pipeline stages. Even though such supergroups are logically independent, and the relevant logic cells around each subgroup drive different PO pins, one can find that they share the same structure. In addition, as discussed in [3], having known that some PI/PO pins can be preplaced on top of their respective connections to reduce PI/PO routing interconnect, a good spatial relation can also be extracted among supergroups to optimize interconnect, despite their logical independence.

In the second stage of extraction, therefore, the closest PI/PO pins are identified that are connected to each supergroup. Given that such PI/PO locations are ideal to interface with other external logic, their physical locations provide clues for relative positioning of supergroups.[3] Fig. 3 shows an example of these supergroups. By taking advantage of the symmetry of logically independent supergroups, fixed pins are found that are connected to each supergroup with the same smallest logical distance and determine a spatial (and total) order of supergroups by referring to these locations.

Table II summarizes supergroups and their cardinalities that are extracted via our techniques on the ISPD 2011 Datapath Placement Benchmarks. On the benchmarks, alignment groups are defined based on net degrees, where alignment groups represent select or data input signals. In the last stage of extraction, it is possible to derive a unique total order on the set G, based on the following definitions and conditions: 1) a supergroup is a totally ordered set of logically related alignment groups; 2) the supergroup order defines a total order of supergroups; and 3) union of all supergroups is the set G.

[3]Graph traversal should precede this stage to define supergroups.

TABLE II
NET DEGREES (DEG.) USED TO DEFINE ALIGNMENT GROUPS, THE NUMBER OF SUPERGROUPS (#*SGs*), AND THEIR CARDINALITIES ($\|SG\|$) ON THE ISPD 2011 DATAPATH PLACEMENT BENCHMARKS

Ckt.	$\vec{d}_k = 0$			$\vec{d}_k = 90$		
	Deg.	No. of SGs	$\|SG\|$	Deg.	No. of SGs	$\|SG\|$
A	68	128	1	257	64	8
B	256	257	1	261	512	1

On structured placement benchmark B, $\|SG\|$ were 1 for both $\vec{d}_k = 0$ and $\vec{d}_k = 90$, which indicates no logical orders were found among any alignment groups.

Section VI-C describes how alignment groups are placed while preserving this extracted order.

VI. STRUCTURE-AWARE GLOBAL PLACEMENT

This section presents techniques for providing alignment constraints to cells during global placement. Skewed weighting with step size scheduling is described and then methods to force alignment by modifying fixed-point locations.

A. Skewed Weighting With Step Size Scheduling

In this section, a skewed weighting technique is introduced that encourages alignment of s_k in preferred placement directions $\vec{d}_k$. The high level idea is to apply gradually increasing net weights for s_k only in orthogonal directions to $\vec{d}_k$. This application of higher net weights in one particular direction (i.e., skewed weighting) increases corresponding costs [HPWL($\vec{x}$) or HPWL ($\vec{y}$)] of the quadratic objective. Consequently, the linear system solver generates compact placement in unpreferred directions to reduce the overall cost.

The rate of change of the weighting value increases slowly during the initial stages of global placement, increases rapidly during the middle stages, and slows again near the end of global placement. This is preferable to applying hard constraints (forced alignment) in the early stage of wirelength optimization, as it can disrupt the original optimization and often lead to a solution that suffers from suboptimality in terms of overall wirelength.

1) *Step Size Scheduling:* Let n be the current global placement iteration and M be its upper bound,[4] and $p(n)$ is defined as the alignment weight schedule function for each iteration n. The function $p(n)$ is adjusted as follows:

$$p(n) = \begin{cases} \dfrac{8n^2}{M^2} & 0 \le n < \dfrac{M}{4} \\ 1 - \dfrac{8\left(n - \dfrac{M}{2}\right)^2}{M^2} & \dfrac{M}{4} \le n \le \dfrac{3M}{4} \\ \dfrac{8(n-M)^2}{M^2} & \dfrac{3M}{4} < n \le M. \end{cases} \quad (4)$$

To avoid imposing hard constraints during the initial stages of global placement, $p(n)$ gradually increases during the initial iterations and to minimize large constraint changes during the final stages, the function decreases toward zero at the last iteration. This function is also used in [28] to model cell density, but it serves a completely different purpose here as a scheduling function. Using $p(n)$, (5) displays the skewed monotonically increasing weighting parameters γ^n and δ^n for alignment net s_k. Using $p(n)$ directly generates very large weighting steps therefore a constant scaling factor β is added. This parameter is left as the default throughout all placement runs. Let $\hat{\mathbf{x}}, \hat{\mathbf{y}}$ be the directional unit vectors and $\sigma^2_{x,y}$ the nth iteration's variance in either the x or y direction. Finally, the modified placement equation is shown in (6). For nonalignment nets, $\delta_{i,j} = 0$ and $\gamma_{i,j} = 0$

$$\gamma^n = \gamma^{n-1} + \hat{\mathbf{y}} \cdot \overrightarrow{\mathbf{d_k}} * \beta * p(n) * \sigma^2_x(n) \quad \text{where } \gamma^0 = 1$$
$$\delta^n = \delta^{n-1} + \hat{\mathbf{x}} \cdot \overrightarrow{\mathbf{d_k}} * \beta * p(n) * \sigma^2_y(n) \quad \text{where } \delta^0 = 1. \quad (5)$$

$$\Phi_G(\vec{x}, \vec{y})^n = \sum_{i,j} [(\gamma^n_{i,j} + w_{i,j})(x_i - x_j)^2 + (\delta^n_{i,j} + w_{i,j})(y_i - y_j)^2]. \quad (6)$$

[4] M is typically set to 50 [13].

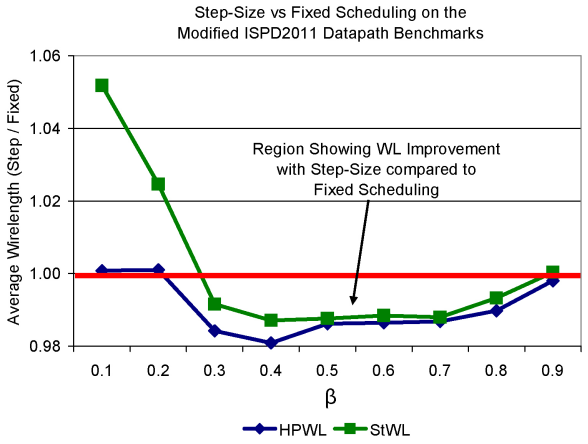


Fig. 4. Average wirelength improvement using the bell-shaped step-size scheduling function on the ISPD 2011 Datapath Benchmark Suite at different values of β.

Fig. 4 displays the average change in wirelength between fixed scheduling and the bell-shaped step-size scheduling using different values of β. It shows that for $0.2 < \beta < 0.9$, the bell-shaped step-size scheduling yields on average, better HPWL as well as StWL. In our implementation, $\beta = 0.5$ is used based on empirical data.

B. Target Skew Ratio Generation

Section VI-A presents a method to systematically increase the skewed weight for an alignment net by either increasing $\gamma_{i,j}$ or $\delta_{i,j}$ as placement evolves. In the preliminary version of this paper [6], the maximum skew for γ^n or δ^n were provided as inputs, primarily based on the designers' discretion. This section avoids this shortcoming by presenting key insights and a method for automatic selection of the maximum skew values. Target skew ratio is defined as follows.

Definition 6: Target skew ratio is the maximum ratio γ^n/δ^n of the net weighting in the x-direction and y-direction for an alignment net during global placement.

Qualitatively, the multiplier value defines how quickly the skewed weight grows with each consecutive global placement iteration. The target skew ratio defines the maximum ratio for a particular net and too large of a target skew negatively impacts the overall wirelength. As an example, given a vertically aligned alignment net, during the initial global placement iteration $\gamma^0 = 1$ and $\delta^0 = 1$. However, by the end of

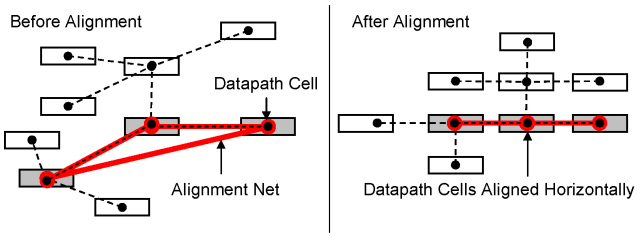


Fig. 5. Horizontal skewed weight force example.

global placement iteration n, the x-direction weight would have increased to $\delta^n = \delta^0 sk_i$ causing the cells within the alignment group to become aligned vertically. Key insights are now outlined for effective target skew multiplier selection and illustrated using Fig. 5. In this figure, the red alignment net is shown connecting each of the datapath cells within an alignment group and the dashed lines indicate net connections among each cell within the alignment group and other cells outside the alignment group.

The first key insight is that the alignment net weight must overcome the interconnect forces imposed by the nets connected among cells within the alignment group. In other words, the force pulling the alignment group into alignment must be greater than the force pulling the cells out of alignment. Equation (7) relates the force from internal pin connections on the alignment net versus the force from external pin connections on the datapath cells

$$T = g^p. \tag{7}$$

This paper solves for the p_i exponent value for a particular alignment net i such that the internal force is always greater than the external force. The variable T equals the total external pin cardinality for an alignment group, which is the sum of all the pins for each cell. This includes pins with a net connecting cells within an alignment group and pins connecting to cells outside of the alignment group. The variable g equals the total number of internal connections which are pins connected to the alignment net. The resulting p value relates the internal connections on the alignment net to the external connections among the cells and a larger p value implies more external connections necessitating a higher skewed weight multiplier. Conversely, a smaller p value indicates fewer external connections thus requiring a smaller multiplier.

The second key insight for automatic target skew ratio selection is consideration of additional interconnect forces from alignment nets themselves.

In the degenerate case, the unpreferred direction retains a weight of zero and this consideration can be ignored. However, for the general case, the unpreferred direction is not zero. The primary reason for a nonzero unpreferred weighting would be to address compaction of the bit stack. Though this paper does not explicitly consider this case, there could clearly be cases where compaction can improve overall placement quality and future paper will explore these options. Thus, the alignment nets are employed to encourage very compact placement in unpreferred directions, however, they pull the connected cells together even in $\vec{d}_k$ when the unpreferred weight is not zero. The HPWL model naturally clumps the cells in alignment nets together in this general case, and therefore the target skew ratio must be properly adjusted to offset this side effect. Our key

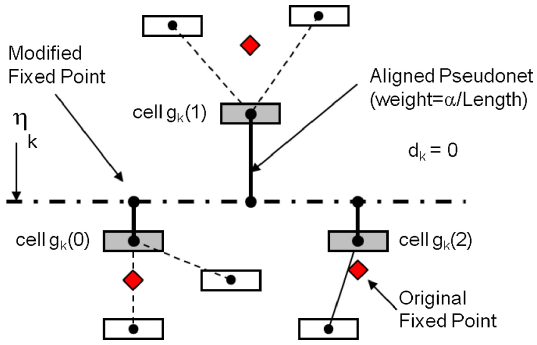


Fig. 6. Example of a fixed-point alignment constraint for a horizontal bit stack. Lookahead legalization generates new zero-area fixed points and the locations of these points are modified to be in alignment with η_k^n.

observation is that the required skew ratio should be linearly increased as the cardinality of an alignment group g_i increases. Thus, the second component of the target skew multiplier is defined as follows: $\alpha * |g_i|, \alpha > 1$ where α is a placer specific constant that insures for a two-pin net, the two cells are always correctly aligned. For example, if aligning two cells horizontally, the Y-weight must be at least α times larger than the X-weight to guarantee the cells align horizontally.[5] For this paper, $\alpha = 1.25$. This constant is then multiplied by the total number of cells in the alignment group. Finally, taking both components into account, the skewed weight multiplier is calculated as shown in

$$\xi_i = (1 + log_g(T)) * \alpha * |g_i|. \tag{8}$$

C. Fixed-Point Alignment

Force-directed global placement frameworks use fixed-points and pseudo nets to discourage cell overlap. By gradually perturbing the unconstrained linear system solver, global placement iterations progressively generate placements with less overlap. In SimPL [13], at each global placement iteration, LAL generates fixed points that are connected to their corresponding movable cells with two-pin pseudo nets. In the subsequent global placement iteration, these pseudo nets exert pulling forces and reduce the amount of cell overlap. For datapath logic, the LAL step and subsequent pseudo net insertion step cause misalignment within the bit stack requiring a constraint forcing alignment which minimizes wrong-way perturbations in the bit stack.

Imposing fixed-point alignment constraint during global placement is achieved in two steps. First, LAL generates fixed-point locations for all movable cells. Second, fixed-point locations are modified for all cells in alignment group g_k. The modified fixed-point location for cell i is denoted as $\eta_{k,i}^n$ for the nth iteration, which is computed as follows:

$$\eta_{k,i}^n = \begin{cases} \left(x_i^n, \dfrac{\Sigma_{j=1,\ldots,|g_k|} y_j^n}{|g_k|} \right), & \text{if } d_k = 0 \\ \left(\dfrac{\Sigma_{j=1,\ldots,|g_k|} x_j^n}{|g_k|}, y_i^n \right), & \text{if } d_k = 90. \end{cases}$$

An example of the modified fixed-point locations and corresponding pseudo nets for a horizontal datapath is shown in Fig. 6. In this example, there are three gray cells, $g_k(0:2)$

[5]This parameter is set when placing only two cells.

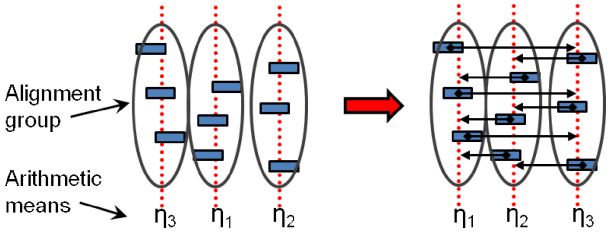


Fig. 7. Imposing ordering constraints by reassigning arithmetic mean values to alignment groups. Subscripts for η_k indicate the extracted order of alignment groups from Section V. Lines with circles at both ends represent modified pseudonet connections.

TABLE III
LEGAL HPWL AND STWL (×10E6) COMPARISON VARYING THE ORDERING CONSTRAINTS USING A FIXED WEIGHTING SCHEME

Ckt.	No Ordering Constraints		Intra-SG Ordering		+Inter-SG Ordering	
	HPWL	StWL	HPWL	StWL	HPWL	StWL
A	12.05	15.79	11.74	15.22	11.46	15.06
B	10.58	15.78	10.58	15.78	10.47	15.71
Avg.	1.03×	1.03×	1.02×	1.01×	1.00×	1.00×

On B, results in columns 2 and 4 and in columns 3 and 5 are identical since no intrasupergroup orders were found.

in one alignment group g_k. The other cell connections are shown with the dashed lines connected to the hollow cells. For random logic cells, the fixed-point locations will be determined solely based on the LAL step. For alignment groups, those locations are modified based on the arithmetic mean parallel to the preferred placement direction $\vec{d}_k$. This modification of fixed-point locations can be further extended to consider ordering constraints, as described next. Imposing ordering constraints across alignment groups is achieved by reassigning the arithmetic mean values to alignment groups. Instead of directly using arithmetic means of fixed-point locations from above: 1) all arithmetic mean locations of alignment groups are collected and sorted by their values, and 2) the sorted arithmetic mean values are assigned to alignment groups following the extracted order of alignment groups from Section V. The assigned arithmetic mean is then used to align fixed-point locations per alignment group (Fig. 7). During early iterations of global placement, this approach may corrupt the spacing among alignment nets if they are not initially uniform. However, as iterations continue, placement evolves locally and recovers good separation among alignment groups as shown in Fig. 8.

Table III empirically demonstrates the impact of imposing ordering constraints on solution quality. Columns 4 and 5 only consider alignment group orders within supergroups and columns 6 and 7 additionally consider the spatial order among supergroups and determines a unique total order. As can be seen from Fig. 8, the order of alignment groups (represented by indices) is preserved during placement.

Modifying the fixed-point locations enables the global placer to progressively reduce cell overlap while maintaining bit-stack alignment. Two items should be noted about this process. First, this paper only modifies the fixed-point locations for datapath logic, not the weighting of the pseudo net. The pseudo-net weighting, separate from the alignment-net weighting described in Section VI-A, acts on datapath and random logic in the same manner. Second, like in other analytical placement algorithms, this technique may temporarily allow cell overlap during global placement, but it progressively reduces with global placement iterations [13, Fig. 6], [27].

VII. STRUCTURE-AWARE DETAILED PLACEMENT

Prior detailed placement techniques can disrupt the structure generated during their wirelength-optimization processes as they are structure unaware. This section presents our two detailed placement techniques to maintain the aligned placement obtained by global placement. The impact of these techniques is reported in Tables VIII, IX, and X in the last row.

A. Bit-Stack Aligned Cell Swapping

This technique extends global cell swapping from [23] for nodes within each g_k by modifying the swap region while keeping the overlap penalty the same. Assuming all cells in the layout region are fixed except for cell j, the swap region, based on the median idea from [10], is the location where the wirelength for cell j is improved if it is swapped with a cell k located in the swap region. This technique seeks cells to swap between the current location of cell j and all cells within the swap region. If swapping improves HPWL, the cell locations are updated.

This paper, unlike [23], bounds the swap region perpendicular to $\vec{d}_k$. Specifically, for each net $p \in E$, the left, right, lower and upper edges of the bounding box are: ($x_l[p]$, $x_r[p]$, $y_l[p]$, $y_u[p]$) and the x^{opt} and y^{opt} from [10] is the median of the x series ($x_l[1]$, $x_r[1]$, $x_l[2]$, $x_r[2]$, ...) and y series ($y_l[1]$, $y_u[1]$, $y_l[2]$, $y_u[2]$, ...), respectively. Because the number of elements is generally even, the x^{opt} and y^{opt} becomes a region with bounding box (x_l^{opt}, y_l^{opt}, x_r^{opt}, y_u^{opt}). Equation (9) finds the modified swap region for a cell within an alignment group with $\vec{d}_k = 0$ while (10) finds it for a cell within an alignment group with $\vec{d}_k = 90$

$$\begin{aligned} &x_l^{\text{opt}}, \min_y(g_k) - \text{var}_y(g_k) \\ &x_r^{\text{opt}}, \max_y(g_k) + \text{var}_y(g_k) \\ &\text{when} \qquad (\overrightarrow{\mathbf{d_k}} = 0) \end{aligned} \tag{9}$$

$$\begin{aligned} &\min_x(g_k) - \text{var}_x(g_k), y_l^{\text{opt}}, \\ &\max_x(g_k) + \text{var}_x(g_k), y_u^{\text{opt}} \\ &\text{when} \qquad (\overrightarrow{\mathbf{d_k}} = 90). \end{aligned} \tag{10}$$

Fig. 9 contrasts the original potential swap region and the structure-aware swap region. In the original example from Fig. 9(a), the swap region for cell j, based on the HPWL would cause j to move out of line with $\vec{d}_k$ for that group, thus disrupting the alignment. In this contrary example, Fig. 9(b) demonstrates that the swap region is shifted down to maintain alignment for that group.

B. Alignment Group Repartitioning

The second detailed placement technique is a top-down recursive repartitioning for each g_k along alignment net s_k, referred to as alignment partitioning shown in Algorithm 2. With datapaths, traditional HPWL metrics can at times fail to detect alignment improvements. This technique minimizes internal net cut values potentially improving overflow without

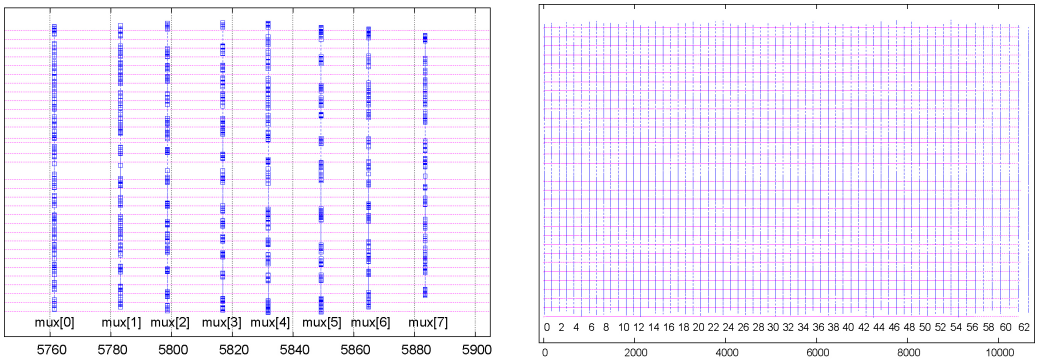


Fig. 8. Placement of (a) MUX select signals within supergroup and (b) MUX[0] signals across supergroups on structured placement benchmark A. Red lines indicate alignment nets with $\vec{d}_k = 0$, while blue dotted lines indicate alignment nets with $\vec{d}_k = 90$. Blue boxes are MUX cells. For both cases, one quarter of the horizontal nets (i.e., alignment nets with $\vec{d}_k = 0$) are shown.

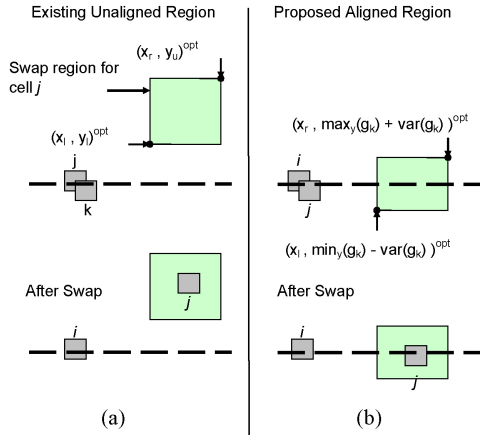


Fig. 9. Swap region shift for cell j when the alignment direction is parallel to the x-axis. The upper y coordinate location is defined by cell i plus the variance between cells i and j. (a) Swap region causes misalignment of the cells. (b) Modified swap region is shown that maintains alignment of the cells.

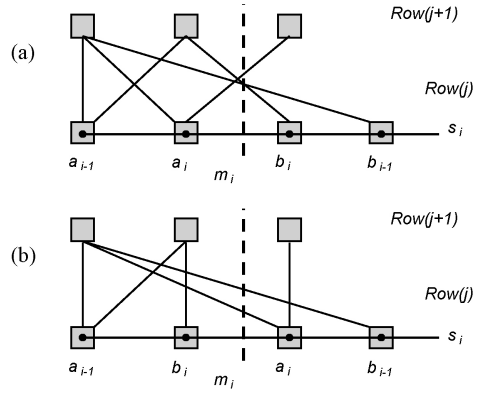


Fig. 10. Group repartitioning example swapping the positions of cell a_i and b_i for an improved net cut. (a) Original cell placement. (b) New placement with improved net cut.

Algorithm 2 Alignment Group Repartitioning

Inputs : A netlist N, alignment groups g_k, alignment nets s_k
Output : A reordered set of nodes g_k along each s_k
function bipartition(g_k):
1: determine median cell within g_k
2: partition g_k into sets A and B with median from 1
3: evaluate initial HPWL of g_k
4: call KL Partitioning with partitions A, B
5: evaluate HPWL $\forall$ nodes $\in A, B$ on updated partition
6: if (HPWL of update partition is $<$ the initial HPWL) g_k = new partition
7: else revert to initial A and B partitions
8: if($|A| >$ minimum partition size) bipartition (A)
9: if($|B| >$ minimum partition size) bipartition (B)
10: return g_k

impacting HPWL and StWL metrics for all nodes in g_k along s_k. The base cut algorithm is from [29], but there are a couple of key differences to the repartitioning method. First, the swap is only accepted when the HPWL after the swap is less than or equal to the HPWL before the swap. Second, the initial median is the midpoint between the nodes. All nodes in g_k with values less than the median go in one partition, while the other nodes go in the other partition.

For each defined alignment group in the design, the algorithm calculates the median or middle point of s_k. Once a median point has been identified, the algorithm partitions all nodes connected to the alignment group using KL partitioning [29]. The partitioning solution is made HPWL-aware by evaluating the KL solution for HPWL changes. If the HPWL increases, the solution is discarded and KL evaluates a new solution for a higher net cut value that does not cause an increase in HPWL or total net cut. Once a partition has been selected, the design is legalized and the loop at that partition level is complete. The algorithm continues to hierarchically break, using recursive repartitioning, each alignment group into smaller partitions until a predefined minimum partition size is met. By minimizing cut value, improved alignment and routability is possible. As an example, consider Fig. 10(a) with median location m_i, alignment group s_i in $Row(j)$. In this placement, the initial cut value across m_i is three. After swapping nodes b_i and a_i, shown in Fig. 10(b), the total net cut value along m_i is reduced to one.

VIII. Experimental Results

Our implementation is in C++, built on the SimPL [13] global placement and FastPlace-DP [23] detailed placement frameworks. It is compatible with the Bookshelf format and requires an additional datapath definition file as input. This file, loaded prior to global placement, includes each cell of the alignment group and the group's direction. Since this paper focuses on the placement solution, each datapath was manually defined for improved experimental control. SAPT is empirically validated with two design styles: 1) the modified ISPD (MISPD) 2011 datapath benchmark suite [19], and

TABLE IV
PARAMETER VALUES SET USED FOR EXPERIMENTS

Parameter	Value	Section	Description
g_k	User defined	III-A	Alignment group
d_k	User defined	III-C	Alignment group direction
β	0.5	VI-A1	Constant scaling factor
α	1.25	VI-B	Constant force required to align two cells

2) industrial hybrid designs. All experiments were run on a 3.2 GHZ Intel(R) Xeon(R) X5672 Linux workstation with 96 GB of memory. Table IV displays parameter settings used for all experiments.

To independently quantify the effectiveness of our global and detailed placement techniques, results are provided for the following three flows.

1) SAPTsg + FP-DP: Only the skewed weighting with step size scheduling technique (Section VI-A) followed by the default FastPlace-DP detailed placer.
2) SAPTgp + FP-DP: Our structure-aware global placement followed by the default FastPlace-DP detailed placer.
3) SAPTgp + SAPTdp: Our structure-aware global and detailed placement techniques.

The SAPT placer is compared against state-of-the-art academic placers:[6] CAPO v10.2 [12], mPL6 [14], NTUPlace3 v7.10.19 [15], Rooster [21], FastPlace v3.0 [16], Dragon v3.01 [17] and SimPL [13]. All placers were supplied a target density of 1.0 to achieve the best wirelength, of which the placers are capable. This configuration corresponds to manual placement, where movable cells are often packed to optimize wirelength with no local whitespace injected. All HPWL and StWL results are reported on legalized placements using the coalesCgrip [20] tool.

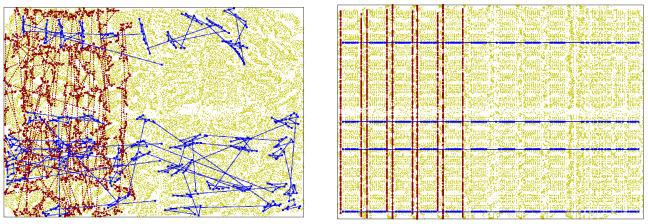
A. Benchmark Circuits

Table V provides the benchmark circuit characteristics. Of note is the number of alignment groups, g_k, and the datapath ratio in each design, where the datapath ratio is defined as the ratio of alignment group cells to random-logic cells. Though the hybrid designs are on the smaller side, they are state-of-the-art industrial circuits, and pose challenges for designers as they contain regular structures intermixed with other random logic. This makes it difficult to place the datapath logic separately from the random logic. The ISPD 2011 Datapath Benchmark Suite contains two datapath circuits, each with eight different utilizations to examine the ability of automatic placers to generate placement solutions at different densities on highly regular structures. In this paper, the benchmarks were modified to make all the latches movable compared to the fixed latch placement in the original work. All logical connections and I/O pin locations remain the same. The SAPT placer is compared against other placers on the Modified ISPD 2011 (MISPD) Datapath Benchmarks, as unfixed latch placement is more challenging and often indicative of hybrid industrial designs.

B. Wirelength Results on the MISPD 2011 Datapath Benchmark Suite

Tables VI and VII compare the total HPWL and total StWL on the MISPD 2011 Datapath Benchmark Suite. To clarify, the total wirelength calculation includes both random-logic and datapath-logic nets in the design. All results are the ratio of the total wirelength of the automatically placed solution to the total wirelength of the manually designed placement as described in [3]. Table VI indicates that the total HPWL of our placement solution is within a few percentage points of the best HPWL solution for each of the benchmarks. In fact, the best HPWL on 7/8 variants for benchmark A are obtained with the SAPT placer.

Table VII demonstrates that the SAPT placer (SAPTgp + SAPTdp) obtains significantly better StWL than other automatic placers for all benchmarks. For the benchmark A variants, the StWL of all the other placers is greater than 1.5× the manually designed solution. For the benchmark B variants, the results are greater than 2.0× the manually designed solution. Alternatively, for benchmark A, the skewed weighting with step step size scheduling technique (SAPTsg + FP-DP) was able to obtain a solution that is 1.35× the manually designed solution at (at 79% utilization). Both global placement techniques (SAPTgp + FP-DP) improved the solution to 1.31× the manually designed solution (also at 79% utilization). Additionally, the detailed placement techniques (SAPTgp + SAPTdp) were able to further reduce the gap to 1.26×. The SAPT placer on benchmark B also significantly outperform prior placers, with SAPTsg + FP-DP achieving 1.49×, SAPTgp + FP-DP achieving 1.48× and SAPTgp + SAPTdp achieving 1.46× the StWL of the manually designed solution. Fig. 11(a) displays the placement solution from SimPL on benchmark A. In this figure, a random selection of cells in alignment groups are plotted with a brown or blue line connecting them. Clearly, the cell placement is not aligned. In the manually placed solution, these cells are aligned, either vertically or horizontally depending on the alignment group, yielding shorter wirelength. Fig. 11(b) shows the placement solution generated using the SAPT placer, with the same set of alignment groups selected as Fig. 11(a), highlighting significant improvement in the alignment of g_k.



(a) (b)

Fig. 11. Forty structured bit stacks are randomly chosen to show the alignment impact of SAPT on benchmark A. (a) Generated by SimPL [13]. (b) Generated by SAPT. Movable cells are shown lightly shaded while the cells in the alignment group are shown dark. Note that cells that are not defined by alignment groups become aligned as well and form (b) regular structure.

C. Wirelength Results on the Hybrid Designs

Table VIII gives the normalized total HPWL and StWL results on the hybrid designs. All results are normalized to the SAPT results (SAPTgp + SAPTdp). As before, SAPT significantly improves StWL on the hybrid designs. Though the HPWL results are similar, SAPT obtains an improvement in StWL between 1% and 13%. For these designs, the StWL improvement is significant, considering the fact that the percentage of datapath logics within the designs is less than 3%.

[6]The recent structure-aware placer from [9] does not provide StWL results for a direct comparison. As of August 2012, a request has been made to the authors for the binary.

TABLE V

CIRCUIT STATISTICS

	ISPD 2011 Datapath Benchmarks		Industrial Hybrid Designs					
	Benchmark A	Benchmark B	Hybrid C	Hybrid D	Hybrid E	Hybrid F	Hybrid G	Hybrid H
Total node count	160 416	152 668	17 922	55 387	83 802	263 906	194 271	62 133
Total pin count	637 984	653 116	64 078	94 682	130 000	397 652	343 727	21 124
Total net count	157 849	148 682	16 874	14 458	16 422	53 884	62 145	101 582
Alignment groups g_k	1425	1932	35	110	60	131	82	28
Datapath ratio	0.920	0.850	0.010	0.012	0.008	0.007	0.018	0.021

Datapath ratio is calculated as the total number of datapath cells divided by the total number of cells.

TABLE VI

TOTAL HPWL RATIO COMPARISON ON THE MISPD 2011 DATAPATH BENCHMARK A AND B VARIANTS ON LEGALIZED PLACEMENTS

	ISPD 2011 Datapath Benchmark A: Total HPWL								ISPD 2011 Datapath Benchmark B: Total HPWL							
Utilization	94	91	89	86	84	82	79	77	95	93	91	89	86	84	81	79
CAPO	**1.05**	1.04	1.04	1.04	1.03	1.02	1.06	1.03	1.20	1.18	1.17	**1.12**	**1.13**	**1.13**	1.14	**1.12**
mPL6	1.17	1.19	1.22	1.14	1.16	1.20	1.17	1.16	1.64	1.86	1.72	1.64	1.65	1.65	1.78	1.78
NTUPlace3	1.22	1.19	1.16	1.19	1.15	1.19	1.23	1.26	1.25	1.19	1.17	1.15	1.16	1.15	**1.12**	1.15
Dragon	1.49	1.58	1.63	1.60	1.51	1.62	1.66	1.60	1.40	1.40	1.35	1.32	1.32	1.30	1.31	1.31
FastPlace3	1.42	1.50	1.53	1.54	1.53	1.67	1.70	1.75	1.69	1.66	1.73	1.71	1.77	1.86	1.77	1.87
Rooster	**1.05**	1.04	1.04	1.04	1.03	1.02	1.06	1.03	**1.15**	**1.15**	**1.13**	**1.12**	1.15	**1.13**	1.13	1.13
SimPL	1.08	1.07	1.06	1.07	1.05	1.06	1.05	1.04	1.23	1.22	1.21	1.20	1.17	1.16	1.16	1.15
SAPTsg + FP-DP	1.09	1.11	1.07	1.05	1.06	1.05	1.04	1.04	1.22	1.20	1.18	1.17	1.17	1.16	1.16	1.15
SAPTgp + FP-DP	1.07	1.04	1.05	1.03	1.03	1.02	1.02	1.01	1.21	1.20	1.17	1.16	1.16	1.16	1.16	1.15
SAPTgp + SAPTdp	1.06	**0.99**	**1.03**	**1.02**	**1.02**	**1.01**	**1.01**	**0.98**	1.21	1.19	1.17	1.16	1.15	1.15	1.14	1.15

Ratios are computed with respect to the manually placed solution.

TABLE VII

TOTAL STWL RATIO COMPARISON ON THE MISPD 2011 DATAPATH BENCHMARK A AND B VARIANTS ON LEGALIZED PLACEMENTS

	ISPD 2011 Datapath Benchmark A: Total StWL								ISPD 2011 Datapath Benchmark B: Total StWL							
Utilization	94	91	89	86	84	82	79	77	95	93	91	89	86	84	81	79
CAPO	1.94	1.94	1.91	1.93	1.90	1.90	1.80	1.90	2.40	2.40	2.38	2.35	2.36	2.36	2.35	2.32
mPL6	2.16	2.14	2.16	2.08	2.10	2.12	2.11	2.09	2.94	3.29	3.06	3.01	2.97	2.95	3.20	3.21
NTUPlace3	2.23	2.18	2.15	2.15	2.11	2.16	2.19	2.09	2.66	2.48	2.47	2.44	2.44	2.44	2.32	2.44
Dragon	2.37	2.44	2.53	2.48	2.36	2.48	2.56	2.43	2.91	2.87	2.84	2.80	2.79	2.77	2.75	2.74
FastPlace3	2.45	2.53	2.56	2.59	2.56	2.71	2.75	2.79	3.73	3.58	3.78	3.79	3.97	4.13	3.96	4.14
Rooster	1.94	1.93	1.91	1.91	1.92	1.91	1.82	1.95	2.40	2.39	2.35	2.33	2.38	2.34	2.34	2.32
SimPL	1.82	1.83	1.80	1.81	1.78	1.78	1.78	1.75	2.27	2.30	2.25	2.24	2.23	2.19	2.24	2.22
SAPTsg + FP-DP	1.43	1.45	1.38	1.37	1.37	1.36	1.35	1.35	1.61	1.57	1.55	1.52	1.52	1.51	1.50	1.49
SAPTgp + FP-DP	1.40	1.35	1.36	1.33	1.33	1.32	1.31	1.31	1.59	1.56	1.54	1.50	1.51	1.50	1.50	1.48
SAPTgp + SAPTdp	**1.35**	**1.28**	**1.31**	**1.31**	**1.28**	**1.28**	**1.26**	**1.28**	**1.58**	**1.55**	**1.52**	**1.49**	**1.49**	**1.48**	**1.48**	**1.46**

The ratios are computed with respect to the manually placed solution. Numbers in bold are the best automated placement results published for these benchmarks.

TABLE VIII

NORMALIZED TOTAL HPWL AND STWL COMPARISON ON THE HYBRID DESIGNS

	Hybrid C		Hybrid D		Hybrid E		Hybrid F		Hybrid G		Hybrid H	
	Total HPWL	Total StWL	Total HPWL	Total StWL	Total HPWL	Total StWL	Total HPWL	Total StWL	Total HPWL	Total StWL	Total HPWL	Total StWL
CAPO	1.13	1.26	1.17	1.32	1.12	1.27	1.19	1.17	1.25	1.23	1.23	1.21
mPL6	1.05	1.15	1.02	1.14	1.20	1.32	1.37	1.30	1.21	1.19	1.09	1.10
NTUPlace3	0.95	1.10	0.95	1.13	0.99	1.19	1.30	1.30	1.02	1.05	1.00	1.01
Dragon	1.10	1.20	2.11	2.04	1.32	1.38	1.29	1.24	–	–	1.22	1.20
Rooster	1.10	1.19	1.16	1.32	1.26	1.23	1.03	1.25	1.03	1.05	1.26	1.23
FastPlace3	0.95	1.04	0.96	1.16	1.22	1.30	1.17	1.14	1.11	1.10	1.04	1.03
SimPL	1.02	1.10	0.97	1.16	1.03	1.12	1.04	1.04	1.01	1.01	1.02	1.02
SAPTsg + FP-DP	1.05	1.08	1.06	1.07	1.07	1.05	1.04	1.05	1.05	1.08	1.08	1.06
SAPTgp + FP-DP	1.02	1.06	1.03	1.03	1.04	1.02	1.03	1.02	1.02	1.04	1.02	1.03
SAPTgp + SAPTdp	1.00	**1.00**	1.00	**1.00**	1.00	**1.00**	1.00	**1.00**	1.00	**1.00**	1.00	**1.00**

The wirelengths are normalized to the SAPT results. The dragon placer was unable to complete for hybrid G.

TABLE IX

TOTAL OVERFLOW ($\times$1E+5) USING THE ROUTER AND EVALUATION SCRIPT FROM THE ISPD 2011 ROUTABILITY-DRIVEN PLACEMENT CONTEST ON THE MISPD 2011 DATAPATH BENCHMARK A AND B VARIANTS ON LEGALIZED PLACEMENTS

	ISPD 2011 Datapath Benchmark A: Routing Overflow								ISPD 2011 Datapath Benchmark B: Routing Overflow							
Utilization	94	91	89	86	84	82	79	77	95	93	91	89	86	84	81	79
CAPO	2.29	2.17	1.72	1.83	1.84	1.68	1.10	2.18	9.16	7.28	7.05	6.68	7.17	7.01	7.13	6.98
mPL6	4.66	4.38	4.44	3.40	3.38	3.65	6.03	5.02	12.7	16.4	14.0	13.6	12.8	12.6	15.3	15.3
NTUPlace3	5.54	5.12	4.63	5.19	4.92	5.63	6.03	5.02	10.2	8.41	8.3	8.09	8.92	9.07	8.21	9.92
Rooster	0.79	1.08	1.22	1.02	1.07	1.64	1.31	1.16	8.11	7.28	6.88	6.72	–	–	–	–
FastPlace3	7.23	8.10	8.72	9.08	8.80	10.4	11.8	12.1	20.8	19.3	21.6	21.7	23.7	25.5	23.5	25.6
SimPL	1.28	1.28	1.22	0.98	0.87	0.87	0.85	0.77	5.98	6.24	5.65	5.49	5.26	4.85	5.21	5.25
SAPTgp + SAPTdp	0.0014	0.038	0.0	0.0	0.0	0.0	0.0	0.0	0.88	0.70	0.55	0.43	0.67	0.58	0.60	0.58

Routing resources are extracted from current 22 nm technology node.

TABLE X

PEAK WEIGHTED CONGESTION (PWC) USING THE BFG-R [30] ROUTER AND EVALUATION SCRIPT FROM THE DAC 2012 ROUTABILITY-DRIVEN PLACEMENT CONTEST ON THE MISPD DATAPATH BENCHMARK A AND B VARIANTS ON LEGALIZED PLACEMENTS

	ISPD 2011 Datapath Benchmark A: Average PWC								ISPD 2011 Datapath Benchmark B: Average PWC							
Utilization	94	91	89	86	84	82	79	77	95	93	91	89	86	84	81	79
CAPO	123.4	123.5	121.3	121.1	121.4	120.3	118.5	128.4	237.4	222.4	222.5	217.4	222.3	220.1	224.3	219.6
mPL6	170.2	188.2	187.3	149.1	145.3	159.9	149.6	139.9	278.2	325.2	313.2	300.0	275.0	315.9	302.9	361.4
NTUPlace3	171.1	170.3	152.1	170.8	166.7	176.9	189.9	173.4	262.5	237.4	239.4	241.4	247.2	247.1	238.6	259.2
Rooster	136.6	163.0	156.5	147.4	152.8	169.4	162.7	174.2	256.2	224.1	217.3	215.1	231.5	221.8	228.7	225.2
FastPlace3	211.9	236.1	237.3	234.7	222.2	224.7	240.2	284.9	341.8	316.2	331.6	334.1	397.6	347.0	341.4	341.7
SimPL	124.7	118.8	117.8	116.2	115.1	114.8	114.1	119.5	207.8	211.3	205.2	205.0	212.5	209.3	204.6	209.2
SAPTgp + SAPTdp	100.4	100.1	100.1	100.1	100.1	100.1	100.1	100.1	151.9	150.7	149.0	148.5	147.9	148.1	149.2	150.1

Routing resources are extracted from current 22 nm technology node.

TABLE XI

TOTAL OVERFLOW (TOF) USING THE ISPD 2011 CONTEST ROUTER (COALESCGRIP [20]) AND THE PEAK WEIGHTED CONGESTION (PWC) USING THE DAC 2012 CONTEST ROUTER (BFG-R [30]) FOR THE HYBRID DESIGNS

	Hybrid C		Hybrid D		Hybrid E		Hybrid F		Hybrid G		Hybrid H	
Routing Metrics	TOF	PWC	TOF	PWC	TOF	PWC	TOF	PWC	TOF	PWC	TOF	PWC
CAPO	1458	107.34	3024	110.46	26 504	117.29	41 520	114.73	131 543	155.56	1186	115.52
mPL6	1360	106.90	940	101.88	23 838	117.02	17 801	111.17	45 904	122.08	490	106.51
NTUPlace3	626	103.66	558	95.92	32 545	133.99	114 843	141.76	42 917	121.33	98	101.36
Rooster	48	100.38	1419	105.22	32 157	122.32	61 731	118.80	134 434	149.26	918	111.49
FastPlace3	372	102.25	722	99.52	77 774	165.88	34 189	118.21	33 723	115.24	30	100.40
SimPL	650	103.31	475	94.60	19 713	118.73	29 524	113.67	37 884	120.33	100	101.11
SAPTgp + SAPTdp	534	103.49	322	93.21	11 276	114.82	28 736	113.66	40 275	119.12	28	100.34

Dragon results omitted as routing was unable to complete. Routing resources are extracted from current 22 nm technology node.

By providing alignment constraints to portions of the datapath, it is observed that other neighboring cells also become aligned during the iterative placement process. The alignment constraints provide hints, directing the placer in the correct gradient. These hints help to overcome local optima, driving placement toward a more globally optimal solution.

D. Routing Congestion Results

To empirically validate the claim that StWL accurately approximates routability, two congestion metrics are reported on all benchmark circuits. It is noteworthy that the SAPT placer is congestion unaware, and that the advantage in routing congestion is simply a by-product of improved placement solutions.

Table IX displays the total overflow (TOF)$\times(1e+5)$, as defined in the ISPD 2011 routability-driven placement contest [30] for the datapath benchmark circuits. All reported overflow values are from the official contest evaluation script. As seen in Table IX, SAPT produces the smallest overflow across all testcases. For benchmark A, SAPT produced a routable placement solution with zero overflow for all but two of the variations. For benchmark B, SAPT improves total overflow by 6.7×, 23.54×, 14.44×, 11.56×, 14.3×, and 10.36× versus SimPL, FastPlace3.0, Dragon, NTUPlace3, mPL6, and CAPO respectively. The second congestion metric is the peak weighted congestion (PWC) as used in the DAC 2012 routability-driven placement contest [31]. This metric calculates the weighted average ACE(x) congestion [32], of the top x% congested g-edges. The smaller the PWC value, the more routable the design. As with the overflow metric, in designs with significant structure, SAPT produces significantly more routable placements than the compared placers.

Table XI displays the TOF using the ISPD 2011 contest router (coalesCgrip [20]) and the PWC using the DAC 2012 contest router (BFG-R [33]) for the hybrid designs. SAPT generated the best PWC results for Hybrids D, E, and G, and was comparable to the best results on the remaining designs. Additionally, SAPT generated the best TOF solution for three Hybrid designs D, E, and H. Though SAPT is not a congestion-aware placer, the significant improvement in routing congestion indicates the strong correlation between StWL quality and congestion.

TABLE XII

COMPARISON OF RUNTIME ON THE HYBRID DESIGNS

Hybrid	C	D	E	F	G	H
CAPO	94.6	74.0	83.4	480.3	482.7	65.9
mPL6	48.5	32.4	36.2	161.7	148.5	35.9
NTUPlace3	13.0	30.0	70.0	278.0	269.0	54.0
Dragon	425.9	193.0	283.9	927.4	–	154.9
Rooster	98.8	84.1	101.1	560.7	455.1	80.9
FastPlace3	13.0	10.7	17.4	55.3	32.93	7.2
SimPL	9.2	12.6	27.1	59.2	40.4	9.2
SAPTgp + SAPTdp	15.9	16.7	38.2	70.9	43.1	10.8

E. Runtime Results

Runtime results on the Hybrid designs are shown in Table XII. Both SimPL and FastPlace3.0 were similar with FastPlace3.0 faster on larger designs. Our SAPT placer performed very competitively compared to the other state-of-the-art placers with the largest design, Hybrid F, taking under 71 seconds to place.

IX. CONCLUSION

Datapath layout was a long-standing challenge in physical design. Many attempts have been made in the last 40 years to close the quality gap between manual and automated placement of datapaths and other regular structures. Nevertheless, a common assumption still prevails among IC designers that circuits with high regularity require manual placement. Our key observation was that the primary optimization objective of modern state-of-the-art placement algorithms–HPWL can mislead placers on datapath-oriented designs. In particular, compressing placement of high-fanout nets to lower the overall HPWL can disrupt the regularity of placement and undermine its Steiner wirelength, which is known to better correlate with

routed wirelength. Based on this observation, a unified framework was developed to enhance current random-logic placers to better handle designs containing datapath logic seamlessly integrating alignment constraints into a state-of-the-art placement engine. Experimental results showed at least a 28% improvement in total StWL compared with the state-of-the-art academic placers for the ISPD 2011 datapath benchmark suite and a 5.8% average improvement in total StWL for industrial hybrid designs. In addition, though not a congestion-aware placer, the techniques showed that significant improvement in routability was achievable through datapath alignment.

A number of open challenges remain with fully automating datapath placement. First, though our comparisons do not report the timing impact because the current implementation is limited to reading the Bookshelf format, significant improvements in wirelength are generally observed to improve timing. Therefore, quantifying timing improvement is an important step for full datapath automation and its evaluation. Second, though this paper shows significant improvements in StWL, only one component of datapath placement (specifically alignment) is addressed. Many datapath styles do not adhere to simple alignment. Other enhancements such as datapath compression could reduce the quality gap between automatic placers and manual placement. Given the assumption that alignment nets and their preferred directions are given in this paper, another important future direction is the automatic extraction of generic datapath logic from a netlist.

ACKNOWLEDGMENT

The authors would like to thank Prof. I. Markov, University of Michigan, and his team for providing the SimPL binary and helpful discussions.

REFERENCES

[1] R. X. T. Nijssen and J. A. G. Jess, "Two-dimensional datapath regularity extraction," in *Proc. ACM/IFIP Workshop Logic Architecture Synthesis*, 1996, pp. 110–117.

[2] P. Ienne and A. GrieBing, "Practical experiences with standard-cell based datapath design tools," in *Proc. ACM/IEEE Des. Autom. Conf.*, Jun. 1998, pp. 396–401.

[3] S. I. Ward, D. A. Papa, Z. Li, C. N. Sze, C. J. Alpert, and E. E. Swartzlander, Jr., "Quantifying academic placer performance on custom designs," in *Proc. ACM Int. Symp. Phys. Des.*, 2011, pp. 91–98.

[4] D. A. Papa, S. N. Adya, and I. L. Markov, "Constructive benchmarking for placement," in *Proc. ACM Great Lakes Symp. VLSI*, 2004, pp. 113–118.

[5] A. B. Kahng and Q. Wang, "Implementation and extensibility of an analytic placer," in *Proc. ACM Int. Symp. Phys. Des.*, 2004, pp. 18–25.

[6] S. I. Ward, M.-C. Kim, N. Viswanathan, Z. Li, C. Alpert, E. E. Swartzlander, Jr., and D. Z. Pan, "Keep it straight: Teaching placement how to better handle designs with datapaths," in *Proc. ACM Int. Symp. Phys. Des.*, 2012, pp. 79–86.

[7] T. Serdar and C. Sechen, "Automatic datapath tile placement and routing," in *Proc. IEEE Des. Autom. Test Eur.*, Mar. 2001, pp. 552–559.

[8] T. Ye, S. Chaudhuri, F. Huang, H. Savoj, and G. D. Micheli, "Physical synthesis for ASIC datapath circuits," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 3. May 2002, pp. 365–368.

[9] S. Chou, M.-K. Hsu, and Y.-W. Chang, "Structure-aware placement for datapath-intensive circuit designs," in *Proc. ACM/IEEE Des. Autom. Conf.*, Jun. 2012, pp. 762–767.

[10] S. Goto, "An efficient algorithm for the two-dimensional placement problem in electrical circuit layout," *IEEE Trans.Circuits Syst.*, vol. 28, no. 1, pp. 12–18, Jan. 1981.

[11] C. Yang, X. Hong, Y. Cai, W. Hou, T. Jing, and W. Wu, "Standard-cell based data-path placement utilizing regularity," in *Proc. IEEE Int. Conf. ASIC*, vol. 1. Oct. 2003, pp. 97–100.

[12] J. A. Roy, D. A. Papa, S. N. Adya, H. H. Chan, A. N. Ng, J. F. Lu, and I. L. Markov, "Capo: robust and scalable open-source min-cut floorplacer," in *Proc. ACM Int. Symp. Phys. Des.*, 2005, pp. 224–226.

[13] M.-C. Kim, D.-J. Lee, and I. L. Markov, "SimPL: An effective placement algorithm," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 31, no. 1, pp. 50–60, Jan. 2012.

[14] T. F. Chan, J. Cong, J. R. Shinnerl, K. Sze, and M. Xie, "mPL6: Enhanced multilevel mixed-size placement," in *Proc. ACM Int. Symp. Phys. Des.*, 2006, pp. 212–214.

[15] T.-C. Chen, Z.-W. Jiang, T.-C. Hsu, H.-C. Chen, and Y.-W. Chang, "A high-quality mixed-size analytical placer considering preplaced blocks and density constraints," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 2006, pp. 187–192.

[16] N. Viswanathan, M. Pan, and C. Chu, "FastPlace 3.0: A fast multilevel quadratic placement algorithm with placement congestion control," in *Proc. Asia South Pacific Des. Autom. Conf.*, 2007, pp. 135–140.

[17] M. Wang, X. Yang, and M. Sarrafzadeh, "Dragon2000: Standard-cell placement tool for large industry circuits," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 2000, pp. 260–263.

[18] T. Kutzschebauch and L. Stok, "Regularity driven logic synthesis," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 2000, pp. 439–446.

[19] S. I. Ward, D. Z. Pan, and E. E. Swartzlander, Jr. (2011). *ISPD Datapath Benchmark Suite* [Online]. Available: http://www.cerc.utexas.edu/utda/download/DP/

[20] H. Shojaei, A. Davoodi, and J. Linderoth, "Congestion analysis for global routing via integer programming," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 2011, pp. 256–262.

[21] J. A. Roy and I. L. Markov, "Seeing the forest and the trees: Steiner wirelength optimization in placement," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 26, no. 4, pp. 632–644, Apr. 2007.

[22] P. Spindler, U. Schlichtmann, and F. M. Johannes, "Kraftwerk2: A fast force-directed quadratic placement approach using an accurate net model," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 27, no. 8, pp. 1398–1411, Aug. 2008.

[23] M. Pan, N. Viswanathan, and C. Chu, "An efficient and effective detailed placement algorithm," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 2005, pp. 48–55.

[24] S. I. Ward, D. Ding, and D. Z. Pan, "PADE: A high-performance mixed-size placer with automatic datapath extraction and evaluation through high-dimensional data learning," in *Proc. ACM/IEEE Des. Autom. Conf.*, Jun. 2012, pp. 768–773.

[25] A. Chowdhary, S. Kale, P. Saripella, N. Sehgal, and R. Gupta, "Extraction of functional regularity in datapath circuits," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 18, no. 9, pp. 1279–1296, Sep. 1999.

[26] A. Rosiello, F. Ferrandi, D. Pandini, and D. Sciuto, "A hash-based approach for functional regularity extraction during logic synthesis," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI*, Mar. 2007, pp. 92–97.

[27] M.-C. Kim and I. L. Markov, "ComPLx: A competitive primal-dual Lagrange optimization for global placement," in *Proc. ACM/IEEE Des. Autom. Conf.*, Jun. 2012, pp. 747–752.

[28] A. B. Kahng, S. Reda, and Q. Wang, "Architecture and details of a high quality, large-scale analytical placer," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 2005, pp. 890–897.

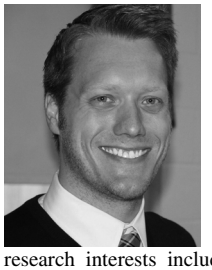
[29] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell Syst. Tech. J.*, vol. 49, no. 1, pp. 291–307, 1970.

[30] N. Viswanathan, C. J. Alpert, C. Sze, Z. Li, G.-J. Nam, and J. A. Roy, "The ISPD-2011 routability-driven placement contest and benchmark suite," in *Proc. ACM Int. Symp. Phys. Des.*, 2011, pp. 141–146.

[31] N. Viswanathan, C. J. Alpert, C. C. N. Sze, Z. Li, and Y. Wei, "The DAC 2012 routability-driven placement contest and benchmark suite," in *Proc. ACM/IEEE Des. Autom. Conf.*, Jun. 2012, pp. 774–782.

[32] Y. Wei, C. Sze, N. Viswanathan, Z. Li, C. J. Alpert, L. N. Reddy, A. D. Huber, G. E. Tellez, D. Keller, and S. S. Sapatnekar, "Glare: Global and local wiring aware routability evaluation," in *Proc. ACM/IEEE Des. Autom. Conf.*, Jun. 2012, pp. 768–773.

[33] J. Hu, J. A. Roy, and I. L. Markov, "Completing high-quality global routes," in *Proc. ACM Int. Symp. Phys. Des.*, 2010, pp. 35–41.



Samuel I. Ward (S'06) received the B.S. and M.S. degrees in electrical engineering from the University of Texas, Austin, in 2004 and 2007, respectively. He is currently pursuing the Ph.D. degree with the University of Texas.

From 2005 to 2012, he was a Circuit Design Engineer with IBM, Austin, TX, working on high-performance microprocessor cores specializing in high-speed datapath circuits. While with IBM, he filed over 20 patents with nine issued, and in 2011 he was named the IBM Master Inventor. His current research interests include physical synthesis of structured nanometer-scale datapaths.

Myung-Chul Kim (M'13) received the Ph.D. degree in electrical engineering from the University of Michigan, Ann Arbor.

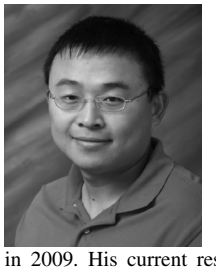
He is currently an Advisory Engineer with IBM, Austin, TX. His current research interests include very large scale integration physical design and synthesis automation with an emphasis on floorplanning, placement, routing, and timing analysis.

Dr. Kim was the winner of the ISPD 2010 Clock-Network Synthesis Contest and the ICCAD 2012 Routability-Driven Placement Contest. He was a recipient of the IEEE/ACM William J. McCalla Best Paper Award at ICCAD 2010.

Natarajan Viswanathan (M'10) received the Ph.D. degree in computer engineering from Iowa State University, Ames, in 2009.

He is currently an Advisory Engineer with the Systems and Technology Group, IBM Corporation, Austin, TX. His current research interests include developing algorithms and methodologies for physical synthesis of nanometer-scale integrated circuits.

Dr. Viswanathan serves on the technical program committees of IEEE ISoCC, IEEE/ACM IWSLIP, and ACM SIGDA Ph.D. Forum at DAC. He was a recipient of the ISPD Best Paper Award, and three Best Paper nominations at DAC and ISPD for his work on integrated circuit placement. He was a recipient of the ACM SIGDA Technical Leadership Award in 2011 for his contributions to the ISPD physical design contests.



Zhuo Li (S'01–M'05–SM'09) received the B.S. and M.S. degrees in electrical engineering from Xi'an Jiaotong University, Xi'an, China, in 1998 and 2001, respectively, and the Ph.D. degree in computer engineering from Texas A&M University, College Station, in 2005.

He is currently a Research Staff Member with the IBM Austin Research Laboratory, Austin, TX. He was one of the co-founders of Pextra Corporation, which was a startup specializing in parasitic extraction and acquired by Mentor Graphics Corporation in 2009. His current research interests include physical synthesis, parasitic extraction, circuit modeling and simulation, and design and analysis of general large-scale algorithms.

Dr. Li is a Committee Member for various conferences in the computer-aided design (CAD) area. He was the Guest Editor of the VLSI Design Journal Special Issue CAD for Gigascale SoC Design and Verification Solutions. He was the Contest Chair of the 2011 and 2012 TAU Power Grid Simulation Contest and is the Co-Chair of the 2012 CAD Contest at ICCAD. He is currently the Secretary of the IEEE Central Texas Section and the Chair of the IEEE CAS/SSC Joint Society Chapter of the Central Texas Section. Recently, the chapter won the IEEE Circuits and Systems Society 2011 Regions 1–7 Chapter of the Year Award and the IEEE Solid State Circuits Society 2011 Outstanding Chapter Award. He is also the Founding Chair of the IEEE CTS CEDA Chapter. He was a recipient of several IBM Technical/Invention Awards, including three IBM Outstanding Technical Achievement Awards. He has filed 35 patents with 8 issued. He has published over 50 conference and journal papers. He was a recipient of the ASPDAC Best Paper Award in 2007, the IEEE Circuits and System Society Outstanding Young Author Award in 2007, the Technical Leadership Award from ACM SIGDA in 2011 for organizing the Power Grid Simulation Contest, and the SRC 2012 Mahboob Khan Outstanding Industry Liaison/Associate Award.

Charles J. Alpert (F'05) received two undergraduate degrees from Stanford University, Stanford, CA, in 1991, and the Doctorate degree in computer science from the University of California, Los Angeles (UCLA) in 1996.

After graduation, he joined the IBM's Austin Research Laboratory, where he currently manages the Design Productivity Group whose mission is to architect design automation tools and methodologies to improve designer productivity and reduce design cost. He has published over 100 conference and journal papers.

Dr. Alpert was a recipient of the Best Paper Award thrice from the ACM/IEEE Design Automation Conference. He has been active in the academic community, serving as the Chair for the Tau Workshop on Timing Issues and the International Symposium on Physical Design. In the last decade, he has served as an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN before stepping down in 2012. He was a recipient of the Mahboob Khan Mentor Award for his work in mentoring in 2001 and 2007.

Earl E. Swartzlander, Jr. (S'64–M'72–SM'79–F'88–LF'11) received the B.S. degree from Purdue University, West Lafayette, IN, in 1967, the M.S. degree from the University of Colorado, Boulder, in 1969, and the Ph.D. degree from the University of Southern California, Los Angeles, in 1972, all in electrical engineering.

He is currently a Professor of electrical and computer engineering with the University of Texas, Austin. He and his students conduct research in application-specific processor design, including high-speed computer arithmetic, embedded processor architectures, and nanotechnology. From 1975 to 1990, he held a variety of positions with TRW, including the Director of independent research and development with the TRW Defense Systems Group. He has written one book and edited seven books. He has authored or co-authored 72 refereed journal papers, 35 book chapters, and 282 conference papers.

Dr. Swartzlander was the Editor-in-Chief of the IEEE TRANSACTIONS ON COMPUTERS from 1990 to 1994. He was the Founding Editor-in-Chief of the *Journal of VLSI Signal Processing*. He has served as an Associate Editor for the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, and the IEEE JOURNAL OF SOLID-STATE CIRCUITS. He has been a member of the Board of Governors of the IEEE Computer Society from 1987 to 1991, the IEEE Signal Processing Society from 1992 to 1994, and the IEEE Solid-State Circuits Council/Society from 1986 to 1991. He has chaired a number of conferences. He was a recipient of the IEEE Third Millennium Medal, the Distinguished Engineering Alumnus Award from the University of Colorado, the Outstanding Electrical Engineer and Distinguished Engineering Alumnus Award from Purdue University, and the IEEE Computer Society Golden Core Award.

David Z. Pan (S'97–M'00–SM'06) received the Ph.D. degree (Hons.) in computer science from the University of California (UCLA), Los Angeles, in 2000.

He is currently an Associate Professor with the Department of Electrical and Computer Engineering, University of Texas, Austin. From 2000 to 2003, he was a Research Staff Member with the IBM T. J. Watson Research Center. He has published over 170 technical papers in refereed journals and conference proceedings. He holds eight U.S. patents.

Dr. Pan has served in the editorial boards of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS (CAS) I and II, *Science China Information Sciences*, the *Journal of Computer Science and Technology*, and IEEE CAS SOCIETY NEWSLETTER. He has also served as the IEEE CAS/CEDA CANDE Committee Chair and the ACM/SIGDA Technical Committee Chair on physical design. He was the General Chair of ISPD 2008 and the Steering Committee Chair of ISPD 2009. He has served on technical program committees of many premier conferences, including the Program Chair of ISPD and the Subcommittee Chair of ASPDAC, DAC, ICCAD, ISLPED, among others. He was a recipient of a number of awards for his research contributions and professional services, including the ACM/SIGDA Outstanding New Faculty Award in 2005, the NSF CAREER Award in 2007, the UCLA Engineering Distinguished Young Alumnus Award in 2009, the SRC Inventor Recognition Award three times, the IBM Faculty Award four times, and nine Best Paper Awards (ASPDAC 2012, ISPD 2011, IBM Research 2010 Pat Goldberg Memorial Best Paper Award in CS/EE/Math, ASPDAC 2010, DATE 2009, ICICDT 2009, and SRC Techcon in 1998, 2007, and 2012). He was a recipient of the Dimitris Chorafas Foundation Research Award in 2000, the eASIC Placement Contest Grand Prize in 2009, the ICCAD'12 CAD Contest Award, the ISPD Routing Contest Award in 2007, and the ACM Recognition of Service Award in 2007 and 2008. He was an IEEE CAS Society Distinguished Lecturer from 2008 to 2009.