Keep it Straight: Teaching Placement how to Better Handle Designs with Datapaths

Samuel I. Ward[†], Myung-Chul Kim[‡], Natarajan Viswanathan[‡],
Zhuo Li[‡], Charles Alpert[‡], Earl E. Swartzlander, Jr.[†], David Z. Pan[†]
[†] ECE Dept. The University of Texas at Austin, Austin, TX 78712
[‡] IBM Austin Research Laboratory, 11501 Burnet Road, Austin, TX, 78758
{wardsi}@utexas.edu, {mckima}@umich.edu, {nviswan, lizhuo, alpert}@us.ibm.com,
{eswartzla}@aol.com, {dpan}@cerc.utexas.edu

ABSTRACT

As technology scales and frequency increases, a new design style is emerging, referred to as hybrid designs, which contain a mixture of random logic and datapath standard cell components. This work begins by demonstrating that conventional Half-Perimeter Wire Length (HPWL)-driven placers under-perform in terms of regularity and Steiner Wire Length (StWL) for such hybrid designs, and the quality gap between manual placement and automatic placers is more pronounced as the designs become more datapath-oriented. Then, a new unified placement flow that simultaneously handles random logic and datapath standard cells is proposed that significantly improves the placement quality of the datapath while leveraging the speed of modern state-of-the-art placement algorithms. The placement flow is built on top of a leading academic force-directed placer. It consists of a series of novel global and detailed placement techniques, collectively called Structure Aware Placement Techniques (SAPT). The techniques effectively integrate alignment constraints into placement, overcoming the deficiencies of the HPWL objective. Experimental results comparing our placement flow with six state-of-the-art placers on the ISPD 2011 Datapath Benchmark Suite show at least a 32% improvement in total StWL with over a $6\times$ improvement in total routing overflow. In addition, the flow demonstrates an 8.25% improvement in total StWL on industrial hybrid designs.

Categories and Subject Descriptors

B.7.2 [**Hardware, Integrated Circuits**]: Design Aids—*Placement and Routing*

General Terms

Design

Keywords

Datapath, Placement, Physical Design

1. INTRODUCTION

As ASIC frequency exceeds $1GHz$ and shrinking schedules drive increased automation for microprocessor designs, the boundary between manually designed datapath logic and random logic macros is blurring. A new design style, referred to as hybrid designs, is emerging that contains 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, with separate placement engines, a dedicated datapath placer may overly constrain the random logic placement solution causing overall degradation in congestion and wire length. A single placement flow handling both structures is extremely valuable, improving design time, 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 is because, in part, current wirelength-driven placement algorithms are unaware of the structure of the datapath. However, with a bit of minimal design guidance, this work shows that an HPWL-driven placer can be taught to handle these situations much better than they do today. The clue is high fanout nets disrupting the structure of the datapath. Most placers will create a clique with a very low weight to model these nets, or treat them just like any other. To minimize the HPWL of a high fanout net, a placer naturally clumps it into a ball, but that is the exact opposite of what is required by a regular datapath structure, as shown in [5].

In this paper, effective techniques are proposed that can be incorporated within existing random-logic targeted placers to better handle designs with embedded datapaths. A novel placement flow is proposed that leverages the speed and flexibility of state-of-the-art HPWL-driven placers while imposing alignment constraints, to achieve better regularity of the datapaths and better StWL results.

The key contributions of this work are as follows:

1. A study of obstacles to current academic placers: the inadequacies and specifically the lack of fidelity of the HPWL model versus StWL model when evaluating 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 2 outlines the problem faced by current random logic placers when placing datapath logic. Section 3 describes the placement flow consisting of two global placement techniques and two detailed placement techniques, which provide alignment constraints to the datapath. Experimental results are presented in Section 4 and finally, conclusions and future work are presented in Section 5.

ISPD'12, March 25–28, 2012, Napa, California, USA.

2. MOTIVATION

Datapath logic can refer to a wide variety of logic circuits including adders, multipliers, rotators, and other logics implemented within pipeline stages. This section discusses: (a) some of the benefits derived from a unified placement framework able to align cells during placement, (b) problems with existing random logic placers, and (c) datapath HPWL accuracy metrics; for designs containing datapath logic circuits.

2.1 The need for a Unified Placement Framework

Datapath logic circuits are traditionally placed by a separate datapath placer such as [1, 6, 7]. These separate datapath placement techniques generate highly efficient and tightly packed placements. After the datapath is placed, these methods generate a larger macro block or small individual bit-slice macro blocks, that are then placed similar to a movable macro block by the main random logic mixed-size placer. The primary drawback of these approaches is that even though a datapath placer may minimize the local wire length through cell ordering [8] or optimizing specific bit-stacks [9], the global connectivity of the placement problem with the embedded datapath is not taken into account. As shown by [2–4], the added constraints from this shortcoming, in general, produce suboptimal results. Additionally, a significant benefit from a unified placement framework comes in the form of reduced development and support costs derived from a single placement framework versus multiple.

2.2 StWL and HPWL Comparisons for Datapath Circuits

Most placers will create a clique with a very low weight to model high fanout nets, or treat them just like any other. To minimize the HPWL of a high fanout net, a placer naturally clumps it into a ball, but that is the exact opposite of what is required by a regular datapath structure, as shown in [5]. The clue to overcoming this problem is to apply alignment constraints during placement.

The way to measure alignment is to use StWL, rather than HPWL to measure the quality of placement, because it more accurately represents routability. To support this point, the following placers mPL6 v6 [11], CAPO v10.2 [12], FastPlace v3.0 [13], NTUPlace3 v7.10.19 [14], Dragon v3.01 [15], and SimPL [16] are compared on both, total Half-Perimeter Wire Length (HPWL) and total Steiner Wire Length (StWL) on the modified ISPD 2011 Datapath Benchmark Suite [10] [1] [3] shown in Table 1. For improved experimental control, all StWL measurements were performed using coalesCgrip [17], and all reported numbers are total wire length results for each design. The HPWL column in Table 1 is sorted from smallest to largest for each benchmark. The table reports both the actual measured HPWL and StWL for the benchmark circuits as well as the wire length ratio compared to the manually placed solution.

Careful examination of this table yields the following surprising results:

1. While HPWL for both benchmarks is very close to the manually placed solution, the StWL results degrade significantly from the manual solution, with the best automated solution at 82% increase in StWL for benchmark A and 227% increase for benchmark B.
2. While fidelity of the HPWL model is expected, for datapath logic it does not hold true. As Table 1 shows, the HPWL column is sorted by increasing value and it is generally expected that StWL would maintain that order. But in fact that does not happen. In both cases, the placer with the best HPWL does not generate the best StWL. For benchmark A, though CAPO generates the best HPWL, 5% larger than the manually placed solution, SimPL generates the best StWL, outperforming CAPO with a 82% increase over the manual solution. The same holds true for benchmark B. Again, CAPO generates the best HPWL, but SimPL generates the best StWL, outperforming CAPO with a 2.43% increase compared to the manual solution.

[1]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/

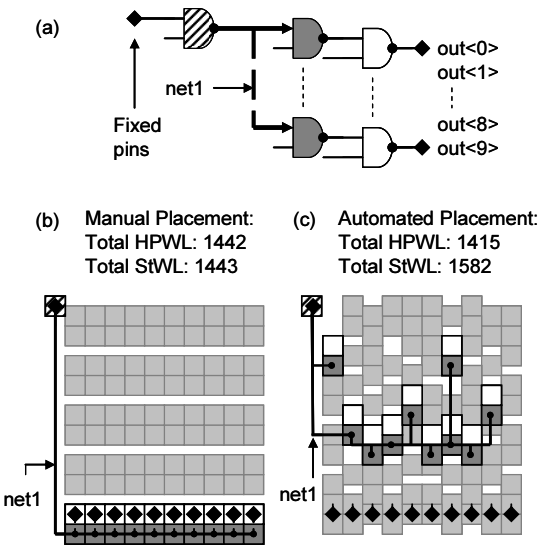


Figure 1: An example circuit where StWL of the manually placed design is better than that of the automated placement, but HPWL of the automated placement solution is better than that of the manual placement. Net1 has fanout of 10.

As Section 4.3 will show, the significant improvement in StWL also corresponds to vastly improved congestion metrics.

2.3 Implicit StWL Optimization through Bit-Stack Alignment

There has been prior work in optimizing StWL directly. As reported in [18], StWL generally correlates with routed wire length (rWL) much better than HPWL. However, optimizing StWL directly during global placement is a hard problem, and iteratively computing StWL can be time consuming. Alternately, this work shows that if a HPWL-driven placer can obtain better alignment for regular structures, it will have better StWL. An example is shown in Figure 1. In Figure 1(a), a partial logic netlist with one NAND gate, shown as hashed, drives net $net1$ with a fanout of 10. All the input and output pins are fixed objects placed on top of the gate. Figure 1(b) shows a manually placed solution for this partial circuit and Figure 1(c) shows a solution from an existing placer. The dark shaded cells match the same dark shaded NAND gates in Figure 1(a). The light shaded grey cells are other logic placed within the design.

For both solutions, we measure the total HPWL and StWL, and the numbers are shown in the figure. As Section 2.2 pointed out, even though the HPWL of the manual solution (1442) is greater than the HPWL of the automated placer (1415), the StWL shows the reverse trend. While it is impractical to list the HPWL and StWL of every single net, clearly for net $net1$, the StWL in Fig-

Table 1: Legalized HPWL and StWL comparison on the ISPD 2011 Datapath Benchmark Suite [10] between manually placed and automated placement solutions. Placement results are sorted by increasing HPWL value. To note: (1) Best HPWL solution does not indicate the best StWL solution. (2) Bold numbers are the best automated placement wire length.

	ISPD Datapath Benchmark A					ISPD Datapath Benchmark B				
	Total HPWL		Total StWL			Total HPWL		Total StWL		
Manually Placed	11000365	1.00	11066683	1.00	Manually Placed	8642097	1.00	9823680	1.00
CAPO	11535525	**1.05**	21516128	1.94	CAPO	10338805	**1.20**	23881606	2.43
SimPL	11837307	1.08	20180311	**1.82**	NTUPlace3	10433894	1.21	26110039	2.66
mPL6	12919955	1.17	23950663	2.16	SimPL	10631304	1.23	22319594	**2.27**
NTUPlace3	13447753	1.22	24673151	2.23	Dragon	12229019	1.42	28577316	2.91
FastPlace3	15672727	1.42	27115750	2.45	FastPlace3	14537026	1.68	36642434	3.73
Dragon	16424739	1.49	26182449	2.37	mPL6	16263018	1.88	28846387	2.94

ure 1(b) is better than the StWL in Figure 1(c). This is due to the better alignment of the whole net in one horizontal row, which produces much better StWL. Also the solution of Figure 1(c) shows the existing placer trying to clump the net into smaller HPWL but causing the StWL to be worse. This paper presents techniques to teach the existing placer to generate a placement solution similar to Figure 1(b) with better StWL than the one in Figure 1(c).

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. Previous works, like post ECO datapath placement, or placing the datapath as a macro block, tend to ignore the connection between random logic and datapath cells since they place every datapath cell in priori. The alignment constraints presented in this work however are providing hints to the placer directing it toward a more globally optimal solution. Thus, as results will show, with relatively few manually defined bit-stacks, the placer generates significantly reduced overall wire length and congestion. The next section outlines the details of supplying these alignment constraints during placement.

3. UNIFIED PLACEMENT WITH ALIGNMENT CONSTRAINTS

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

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

$$HPWL(\vec{x}) = \sum_{e \subset E} [MAXx_i - MINx_i] \quad (2)$$

Modern placers often approximate HPWL by a differentiable function using the quadratic objective, defined as:

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

From Equation 3, (x_i, y_i) represents the coordinates of cell i, and $w_{i,j}$ represents the weight between cells i and j. In this work, a force-directed global placer in the spirit of SimPL [16], where $w_{i,j}$ is given by the Bound2Bound net model [19], is used along with a detailed placer similar to FastPlace-DP [20]. Briefly, SimPL is a flat, force-directed global placer. It maintains a lower-bound and an upper-bound placement and iteratively narrows the displacement between the two to yield a final placement solution. The upper-bound placement is generated by applying lookahead legalization, which is based on top-down geometric partitioning and non-linear scaling. The coordinates obtained from the upper bound placement are used to generate the fixed-points and pseudo nets for force-directed placement. The lower-bound placement is then generated by minimizing the quadratic objective in Equation 3.

In this paper, it is assumed a set of T datapath groups and their directions are given. Each datapath group $g_k \in G$, $0 < k < T$, is an unordered nonoverlaping subset of cells from V. Generally, each g_k corresponds to the bit-stack 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 for structured latch placement. The direction of datapath g_k is defined as $\vec{d_k}$. In this paper, only horizontal and vertical directions are considered, which means $\vec{d_k} \in (0, 90)$. In the example shown in Figure 1, $\vec{d_k} = 0$.

The above assumptions that the datapath groups and directions are given are valid and practical. One may use datapath extractors such as [21, 22], based on circuit properties, to generate the datapath. This information could also be provided by designers, which may come directly from the logic description of the nelist, or designer experience. As an example, if a designer is trying to structure the latch placement to be vertical, it is trivial for him to provide the vectored latch name and the direction (horizontal or vertical).

3.1 Alignment Nets

Adding pseudo nets is a common method used by modern placers to provide spreading forces. In this work, similar to SimPL, during every iteration of global placement, pseudo nets are added after lookahead legalization to enforce spreading during the subsequent linear system solver.

Definition 1. 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 $\alpha \cdot w_{i,j}$, defined in [16], and does not exist in the circuit netlist.

It shall be noted 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 increasing parameter α is described in [16], 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 α value and gradually increasing through each iteration. This scheme provides flexibility to the placer during the early stages, while tightening the constraints for no overlap towards the end of global placement.

To generate better datapath alignment, one approach is direct manipulation of existing nets between the datapath cells. But this approach interferes with other prior placement enhancements. Specifically, direct weighting manipulation of current nets disrupts timing aware placement and net weighting for those cells. Due to the above problem, a new method is instead proposed. A new category of nets, refereed as alignment nets is defined.

Definition 2. An alignment net s_k, where $0 < k < T$ and $T = |G|$, is a weighted multi-pin connection between all cells in the

datapath group g_k. For placement, the alignment net is modeled using the Bound2Bound net model [19].

These nets are created at the beginning of global placement and remain persistent during the entire global and detailed placement stages. A skewed net-weight schedule (Section 3.3), helps these nets align the cells within the corresponding datapath group g_k inside the placement region. By applying the alignment constraints to a new net s_k, prior techniques continue to function as before.

3.2 Unified Placement Flow Overview

The proposed new placement flow is presented in Figure 2, where the shaded boxes highlight the enhancements applied to each g_k. During global placement, after pseudo net insertion for all cells, the modified flow applies the *skewed weighting with step size scheduling* to each datapath group. Then after the linear solver and fixed-point generation, the second global placement modification is applied called *fixed-point and pseudo net alignment constraint*. Once the global placement solution has converged as defined in [16], two detailed placement steps that act only on the datapath logic are presented. The first is *bit-stack aligned cell swapping* and the second is *datapath group repartitioning*. Detailed placement for the random logic cells is implemented using the techniques presented in [20]. At each step, the modifications apply only to the defined placement groups g_k leaving all other random logic cells to be placed as they would before. Though in this work SimPL and FastPlace-DP are used as an example, the techniques can be adapted for other force directed global placement and detail placement methods as well.

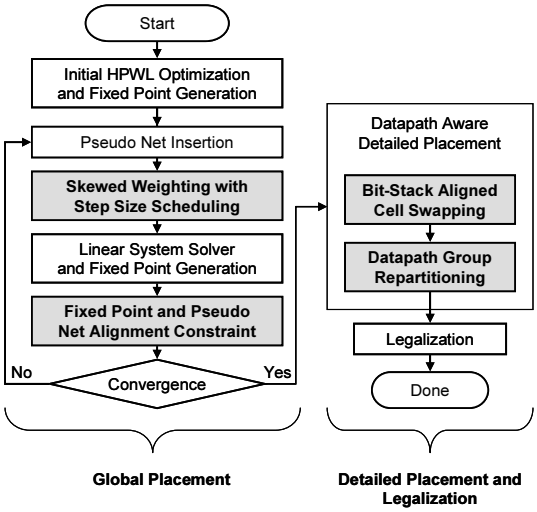


Figure 2: Proposed unified datapath-aware placement flow. The baseline components are shown in transparent boxes and the added datapath-aware components are shaded.

3.3 Skewed Weighting with Step Size Scheduling

In this section, a skewed weighting process applied to each alignment net s_k is described that improves alignment along the datapath. The high level idea is to add a skewed weight for each datapath group, with the weight gradually increasing during each iteration. 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.

Applying hard constraints (forced alignment) in the early stage of wire length optimization can disrupt the original optimization and often can lead to a solution that suffers from sub-optimality in terms of overall wire length.

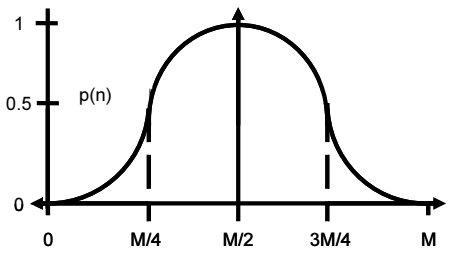


Figure 3: Bell-shaped step size scheduling function.

Thus, let n be the global placement iteration number and M be its upper bound[2] and define $p(n)$ as the alignment weight schedule function for each iteration n. The following equation for $p(n)$ is proposed:

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

To minimize 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 [23] as a penalty function, but it serves a *completely different purpose* here as a scheduling function. Using $p(n)$, Equation 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 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 Equation (6). For non-alignment nets, $\delta_{i,j} = 0$ and $\gamma_{i,j} = 0$.

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

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

3.4 Fixed-Point Alignment Constraint

Modern force-directed global placement frameworks use fixed-points and pseudo nets to increase spreading. By gradually perturbing the unconstrained linear system solver, consecutive placement solutions with less overlap are generated In SimPL [16], after each global placement iteration, lookahead legalization generates a fixed zero-area anchor with two-pin pseudo nets. During the following global placement iteration, these pulling forces reduce the amount of cell overlap. For datapath logic, the lookahead legalization 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.

The proposed fixed-point alignment constraint is applied in two steps. First, lookahead legalization generates a fixed-point location for all cells. Second, for all cells in datapath group g_k, a modified fixed-point is added. The defined location of this fixed-point for

[2] M is typically upper-bounded by 50 [16]

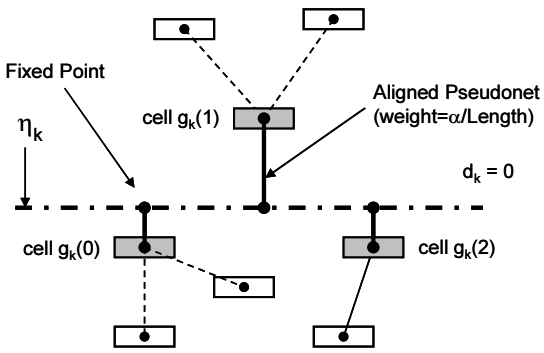


Figure 4: Example of a fixed-point alignment constraint for a horizontal bit-stack. Lookahead legalization generates new zero area fixed-points and the locations of these points are modified to be in alignment with η_k^n.

cell i as $\eta_{k,i}^n$ for the nth iteration. In this paper, $\eta_{k,i}^n$ is computed as follows:

$$\eta_{k,i}^n = (x_i, \sqrt[|g_k|]{\prod_{j=1,\ldots,|g_k|} y_j}), \quad \text{if } d_k = 0$$
$$\eta_{k,i}^n = (\sqrt[|g_k|]{\prod_{j=1,\ldots,|g_k|} x_j}, y_i), \quad \text{if } d_k = 90 \tag{7}$$

An example of the modified fixed-point locations and corresponding pseudo nets for a horizontal datapath is shown in Figure 4. In this example, the three grey cells, $g_k(0:2)$ are in one datapath group g_k. The other cell connections are shown with the dashed line connected to the hollow cells. For random logic cells, the fixed-point will be determined based on the lookahead legalization step alone. For the datapath cells shown, after lookahead legalization generates a new fixed-point location, those locations are modified based on the geometric mean parallel to the datapath direction $\vec{d}_k$.

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 work *only modifies the fixed-point location* for datapath logic, not the weighting of the pseudo net. The pseudo net weighting, disparate from the alignment net weighting proposed in Section 3.3, acts on datapath and random logics the same. Second, though this technique violates the overlap constraint during global placement, the overlap reduces with consecutive global placement iterations [16] and small overlaps can be easily removed during legalization without undermining the overall wire length.

3.5 Bit-Stack Aligned Cell Swapping

This detailed placement technique modifies global cell swapping from [20] for nodes within each g_k by modifying the "swap region" while keeping the overlap penalty the same. Assuming all cells in the placeable region are fixed except for cell j, the "swap region", based on the median idea from [8], is the location where the wire length for cell j is improved if it is swapped with a cell k located in the swap region. This technique looks for cells to swap between the current location of cell j and all cells within the swap region. If a swap produces improved HPWL, the cell locations are updated.

This work, unlike [20], bounds the swap region perpendicular to $\vec{d}_k$. More 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 [8] is the median of the x series ($x_l[1]$, $x_r[1]$, $x_l[2]$, $x_r[2]$, ...) and y series ($y_l[1]$, $y_u[1]$, $y_l[2]$, $y_u[2]$, ...).

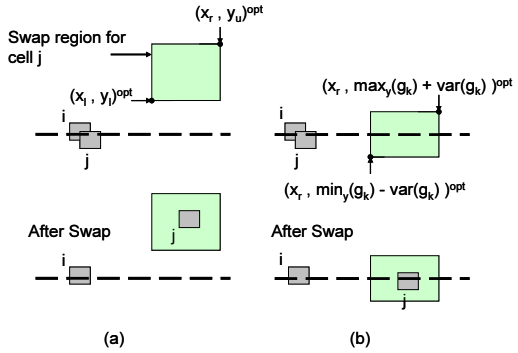


Figure 5: Swap region shift for cell j when the datapath direction is parallel to the x-axis. The upper y coordinate location is defined by cell i plus the variance between cell i and cell j.

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}). The modified swap region assuming the alignment net s_k is parallel to the x-axis is shown in Equation 8, and assuming the alignment net s_k is parallel to the y-axis is shown in Equation 9.

$$\begin{aligned} & x_l^{opt}, min_y(g_k) - var_y(g_k) \\ & x_r^{opt}, max_y(g_k) + var_y(g_k) \\ & when \qquad (\overrightarrow{\mathbf{d_k}} = 0) \end{aligned} \tag{8}$$

$$\begin{aligned} & min_x(g_k) - var_x(g_k), y_l^{opt}, \\ & max_x(g_k) + var_x(g_k), y_u^{opt} \\ & when \qquad (\overrightarrow{\mathbf{d_k}} = 90) \end{aligned} \tag{9}$$

Figure 5 Illustrates the difference between the original potential swap region and the datapath aware swap region. In the original example from Figure 5(a) on the left, 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 the proposed method shown in Figure 5(b) on the right, the swap region is shifted down to maintain alignment for that group.

3.6 Datapath 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 datapath partitioning. 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. The partitioning iterates through each node in g_k and swaps among other cells in g_k that minimize the total net cut of that partition. By minimizing cut value, improved alignment and routability is possible. The base cut algorithm is from [24], 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, the other nodes in the other partition.

For each defined datapath 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 datapath alignment group using KL partitioning. 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

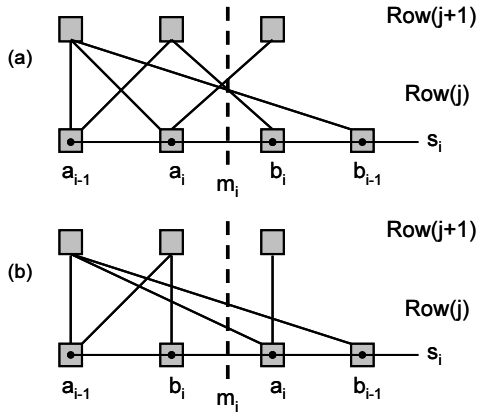


Figure 6: Group repartitioning example swapping the positions of cell a_i and b_i for an improved net cut.

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 datapath group into smaller partitions until a predefined minimum partition size is met. As an example, consider Figure 6(a) with median location m_i, datapath 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 Figure 6(b), the total net cut value along m_i is reduced to one.

4. EXPERIMENTAL RESULTS

The placer in this work is compatible with the Bookshelf format and requires an additional datapath definition file as input. This file is loaded prior to global placement, and includes each manually identified datapath groups to be aligned and provides a direction for each datapath group. For improved experimental control, all HPWL numbers and StWL estimates were generated using coalesCgrip [17]. The ISPD 2011 Datapath Benchmark Suite for the manual placement and spacing variations were modified to make all latches movable compared to the fixed latch placement in the original work. All logical connections and input and output pin locations remained the same. Briefly, the ISPD 2011 Datapath Benchmark Suite [10] contains two common datapath circuits each with a series of eight different utilizations to examine the ability of automatic placers to generate placement solutions at different densities.

All placement numbers are the ratio of the *total wire length* of the placed solution vs. the *total wire length* of the manually designed placement as described in [3]. All wire length numbers are from legalized placements and in cases where overlaps were generated, post placement legalizers were used to generate a legal placement. This work focuses on the placement solution, so each datapath was manually defined for improved experimental control.

The proposed approach is referred to as Structure Aware Placement Techniques (SAPT). In the tables that follow, *SAPTgp* refers to the proposed structure aware global placement techniques with the base FastPlace-DP [20] detailed placer, and *SAPTdp* refers to the wire length results when running both the proposed global and detailed placement techniques. All placers were supplied a target density requirement of 1 as defined in the ISPD placement contests [25]. The placer ROOSTER [18], a variant of Capo that optimizes StWL in global and detailed placement, was also run and we observed slightly improved HPWL and StWL results with little impact on overflow numbers when compared to CAPO.

Table 2 provides benchmark characteristics. Of note is the number of datapath groups g_k in each design and the datapath ratio. The datapath ratio is defined as the ratio of datapath cells to random logic cells in each design. Though the hybrid designs are on the smaller side, they are state-of-the-art industrial circuits containing both datapath and random logic cells. They are included to demonstrate the impact of the proposed techniques on modern designs. Due to page limitations, run times are reported for only the hybrid industrial designs.

4.1 Results on the Modified ISPD2011 Datapath Benchmark Suite

Table 3 shows the total HPWL ratio (including both random and datapath logic nets in the design) for the ISPD2011 Datapath Benchmark Suite comparing prior placers to the manually generated layout solutions. In these runs for benchmark A, CAPO generated the best total HPWL results for all placers coming within 2% of the manually placed benchmark at 82% design utilization. The SimPL placer also generated very competitive HPWL results at only 4% increase at 77% utilization. NTUPlace3 failed to run on benchmark A. For benchmark B, both NTUPlace3 and CAPO generated the best overall total HPWL result among all placers at 12% worse than the manual solution. For all proposed techniques, the HPWL numbers oscillate within a few percentage points of the original placement solution from SimPL and FastPlace-DP detailed placement. As will be shown, though the total HPWL numbers are approximately the same as the manually placed solution, the total StWL of the automated placers is significantly worse. This result reinforces that for datapath designs, total HPWL is a bad indicator of placement quality.

Total steiner wire length (StWL) results (including all nets in the design) for each datapath benchmark A and B variant are shown in Table 4. As previously shown, total StWL results of prior placement algorithms were abysmal compared to the total StWL of the manually placed benchmark. However, the proposed global placement (SAPTgp) solutions improved the StWL from 1.78 to 1.35 compared to the manual solution and the detailed placement methods (SAPTdp) further improves the ratio from 1.78 to 1.30. The proposed placer on benchmark B also significantly outperformed prior placers with SAPTgp achieving 1.48 and SAPTdp achieving 1.46 compared to the manual designed solution. The results in bold represent the best published automated StWL placement solution.

These results show that the presented placer for benchmark A outperforms all other automated placers with SimPL coming closest at a 36% increase compared to SATPdp. For benchmark B, the proposed placer outperforms all other placers with SimPL again being the closest at a 48% increase over SAPTdp.

Figure 7 (a) displays the datapath placement solution from SimPL. In this figure, a random selection of bit-stack cells are plotted with a black line connecting them. In the manually placed solution, the bit stack is aligned, either vertically or horizontally depending on the group, which allows the placer to obtain a more compact placement solution. As shown in Figure 7 (a), clearly these cells are not placed in alignment. The modified placement solution generated using the proposed placer is shown in Figure 7 (b). The same set of datapath groups shown in Figure 7 (a) are displayed with a black line connecting each cell. Clearly there is significant straightening improvement in each bit-stack group.

4.2 Hybrid Placement Results

In addition to significantly improved StWL results on the datapath benchmarks, the proposed placer generates improved hybrid design StWL results as shown in Table 5. For each placer, the num-

Table 2: Circuit statistics. Datapath ratio is calculated as the total number of datapath cells divided by the total number of cells.

	ISPD2011 Datapath Benchmarks		Industrial Hybrid Designs			
	Benchmark A	Benchmark B	Hybrid C	Hybrid D	Hybrid E	Hybrid F
Total node count	160416	152668	17922	55387	83802	263906
Total pin count	637984	653116	64078	94682	130000	397652
Total net count	157849	148682	16874	14458	16422	53884
Datapath groups g_k	1425	1932	35	110	60	131
Datapath ratio	0.920	0.850	0.010	0.012	0.008	0.007

Table 3: Total HPWL ratio comparison on the modified ISPD 2011 Datapath Benchmark A and B variants with legalized placement. The ratios are computed with respect to the manually placed solution.

	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
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
SAPTgp	1.10	1.12	1.07	1.05	1.06	1.05	1.04	1.04	1.21	1.20	1.17	1.16	1.16	1.16	1.16	1.15
SAPTdp	1.09	1.07	1.05	1.05	1.04	1.04	1.03	1.04	1.21	1.19	1.17	1.16	1.15	1.15	1.14	1.15

Table 4: Total StWL ratio comparison on the modified ISPD 2011 Datapath Benchmark A and B variants with *unfixed latches* after legalized placement. The ratios are computed with respect to the manually placed solution. Numbers in bold are the best automated placement results published for these benchmarks.

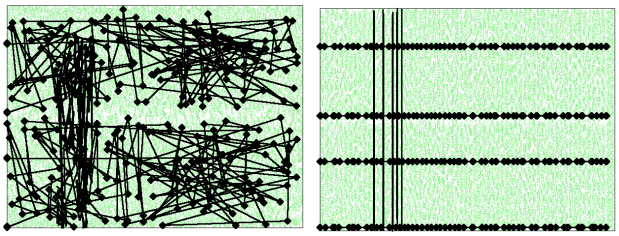
	ISPD 2011 Datapath Benchmark A: Total StWL								ISPD 2011 Datapath Benchmark B: Total StWL							
Utilization	94	91	89	86	84	82	79	77	95	93	91	89	86	84	81	79
CAPO	1.94	1.94	1.91	1.93	1.90	1.90	1.80	1.90	2.40	2.40	2.38	2.35	2.36	2.36	2.35	2.32
mPL6	2.16	2.14	2.16	2.08	2.10	2.12	2.11	2.09	2.94	3.29	3.06	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
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
SAPTgp	1.43	1.46	1.39	1.36	1.37	1.36	1.35	1.35	1.59	1.56	1.54	1.50	1.51	1.50	1.50	1.48
SAPTdp	1.38	1.35	1.34	1.34	1.32	1.32	**1.30**	1.32	1.58	1.55	1.52	1.49	1.49	1.48	1.48	**1.46**

Table 5: The impact of datapath placement on hybrid designs. The wire length ratios are compared to the proposed placer.

	Hybrid C			Hybrid D			Hybrid E			Hybrid F		
	Total HPWL	Total StWL	Run Time(s)	Total HPWL	Total StWL	Run Time(s)	Total HPWL	Total StWL	Run Time(s)	Total HPWL	Total StWL	Run Time(s)
CAPO	1.13	1.26	94.6	1.17	1.32	74.0	1.12	1.27	83.4	1.19	1.17	480.3
mPL6	1.05	1.15	48.5	1.02	1.14	32.4	1.20	1.32	36.2	1.37	1.30	161.7
NTUPlace3	0.95	1.10	13.0	0.95	1.13	30.0	0.99	1.19	70.0	1.30	1.30	278.0
Dragon	1.10	1.20	425.9	2.11	2.04	193.0	1.32	1.38	283.9	1.29	1.24	927.4
FastPlace3	0.95	1.04	13.0	0.96	1.16	10.7	1.22	1.30	17.4	1.17	1.14	55.3
SimPL	1.02	1.10	9.2	0.97	1.16	12.6	1.03	1.12	27.1	1.04	1.04	59.2
SAPTdp	1.00	1.00	15.9	1.00	1.00	16.7	1.00	1.00	38.2	1.00	1.00	70.9

Table 6: The Total Overflow (x 1e+5) using the router and evaluation script from the ISPD 2011 routability-driven placement contest on the modified ISPD 2011 Datapath Benchmark A and B variants with unfixed latches after legalized placement. The "Total Overflow", a measure of the routing congestion of the placement solution, is reduced to zero on six of the benchmark A variants and reduced by at least 6.7x for all benchmark B variants.

	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
Dragon [3]	-	-	-	-	-	-	-	-	12.8	12.7	12.5	12.4	12.6	12.7	12.8	12.9
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	0.0012	0.032	0.0	0.0	0.0	0.0	0.0	0.0	0.90	0.70	0.56	0.45	0.48	0.59	0.62	0.59
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



(a) (b)

Figure 7: Forty structured bit-stacks are randomly chosen to show the impact of the proposed placer on structured nets in Benchmark A. (a) is generated by SimPL [16] whereas (b) is generated by the proposed placer. Movable cells are shown lightly shaded while the bit-stack connectivity is shown in black.

bers indicate the ratio of the total wire length obtained by the placer to that obtained by our techniques (given under SAPTdp). Though the HPWL results are similar, the proposed placer obtains an improvement in StWL between 4% and 13%. For these designs, the wire length improvement is significant considering the percentage of datapath logics within the designs is less than 1.5%.

By providing alignment constraints to portions of the datapath, it is observed that neighboring cells also get 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 towards a more globally optimal solution. Thus, with relatively few manually pre-defined bit-stacks, this work shows that a HPWL-driven placer can generate improved solutions for the other cells, resulting in significantly improved total wire length.

4.3 Routing Congestion Results

To empirically prove our claim that StWL accurately approximates routability, Table 6 displays the total overflow ($\times 1e+5$), as defined in the ISPD 2011 routability-driven placement contest [26]. Reported overflow numbers are provided using the contest evaluation script on legal placements. As seen in Table 6, SAPT produces the smallest overflow across all test cases. For benchmark A, SAPT produced a routable placement solution with zero (0) overflow for all but two of the variations. For benchmark B, SAPT improves total overflow by $6.7\times$, $23.54\times$, $14.44\times$, $11.56\times$, $14.3\times$, and $10.36\times$ versus SimPL, FastPlace3, Dragon, NTUPlace3, mPL6, and CAPO respectively. Though SAPT is not a congestion aware placer, the significant improvement in routing congestion indicates the strong correlation between alignment, congestion and the importance of StWL for datapath logics.

5. CONCLUSIONS

This work presents a unified framework to enhance current random logic placers to better handle designs containing datapath logics. A set of new global and detail placement techniques, including skewed weighting with step size scheduling, fixed-point and pseudo net alignment constraint, bit-stack aligned cell swapping and group recursive repartitioning, were presented that seamlessly integrate alignment constraints into a state-of-the-art placement engine to overcome the shortcomings of the HPWL model for datapaths. Experimental results show at least a 32% improvement in total StWL compared with six state-of-the-art academic placers for the ISPD 2011 Datapath Benchmark Suite and a 8.25% average improvement in total StWL over six state-of-the-art placers for industrial hybrid designs. Though comparisons do not report the timing impact because the current implementation is limited to reading the Bookshelf format, significant improvements in wire length are generally attributed to improved timing. Future research will including quantifying this effect in addition to automatically extracting the datapath.

6. REFERENCES

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

[2] P. Ienne and A. GrieBing, "Practical experiences with standard-cell based datapath design tools," in *Proceedings of DAC*, pp. 396–401, 1998.

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

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

[5] T. Kutzschebauch and L. Stok, "Regularity driven logic synthesis," in *Proc. ICCAD*, pp. 439–446, 2000.

[6] T. Serdar and C. Sechen, "Automatic datapath tile placement and routing," in *Design, Automation and Test in Europe, 2001. Conference and Exhibition 2001. Proceedings*, pp. 552 –559, 2001.

[7] T. Ye, S. Chaudhuri, F. Huang, H. Savoj, and G. D. Micheli, "Physical synthesis for ASIC datapath circuits," in *Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on*, vol. 3, pp. III–365 – III–368 vol.3, 2002.

[8] S. Goto, "An efficient algorithm for the two-dimensional placement problem in electrical circuit layout," *Circuits and Systems, IEEE Transactions on*, vol. 28, pp. 12 – 18, jan 1981.

[9] C. Yang, X. Hong, Y. Cai, W. Hou, T. Jing, and W. Wu, "Physical synthesis for ASIC datapath circuits," in *ASIC, 2003. Proceedings. 5th International Conference on*, vol. 1, pp. 97–100, 2003.

[10] S. I. Ward, D. Z. Pan, and E. Swartzlander, "2011 ISPD Datapath benchmark suite." http://www.cerc.utexas.edu/utda/download/DP/, Mar 2011.

[11] T. F. Chan, J. Cong, J. R. Shinnerl, K. Sze, and M. Xie, "mPL6: enhanced multilevel mixed-size placement," in *Proc. ISPD*, pp. 212–214, 2006.

[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. ISPD*, pp. 224–226, 2005.

[13] N. Viswanathan, M. Pan, and C. Chu, "FastPlace 3.0: A fast multilevel quadratic placement algorithm with placement congestion control," in *Proceedings of ASPDAC*, pp. 135–140, 2007.

[14] 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. ICCAD*, pp. 187–192, 2006.

[15] M. Wang, X. Yang, and M. Sarrafzadeh, "Dragon2000: Standard-cell placement tool for large industry circuits," in *Proc. ICCAD*, pp. 260–263, 2000.

[16] M.-C. Kim, D.-J. Lee, and I. L. Markov, "simPL: an effective placement algorithm," in *Proc. ICCAD*, pp. 649–656, 2010.

[17] H. Shojaei, A. Davoodi, and J. Linderoth, "Congestion analysis for global routing via integer programming," in *Proc. ICCAD*, pp. 256–262, 2011.

[18] J. A. Roy and I. L. Markov, "Seeing the forest and the trees: Steiner wirelength optimization in placement," *CAD of Integrated Circuits and Systems, IEEE Transactions on*, vol. 26, no. 4, pp. 632–644, 2007.

[19] P. Spindler, U. Schlichtmann, and F. M. Johannes, "Kraftwerk2 - a fast force-directed quadratic placement approach using an accurate net model," *IEEE TCAD*, vol. 27, no. 8, pp. 1398–1411, 2008.

[20] M. Pan, N. Viswanathan, and C. Chu, "An efficient and effective detailed placement algorithm," in *Proc. ICCAD*, pp. 48–55, 2005.

[21] A. Chowdhary, S. Kale, P. Saripella, N. Sehgal, and R. Gupta, "Extraction of functional regularity in datapath circuits," *IEEE TCAD*, vol. 18, no. 9, pp. 1279–1296, 1999.

[22] A. Rosiello, F. Ferrandi, D. Pandini, and D. Sciuto, "A hash-based approach for functional regularity extraction during logic synthesis," in *Proc. ISVLSI*, pp. 92–97, 2007.

[23] A. B. Kahng, S. Reda, and Q. Wang, "Architecture and details of a high quality, large-scale analytical placer," in *Proc. ICCAD*, pp. 890–897, 2005.

[24] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell Systems Technical Journal*, vol. 49, no. 1, pp. 291–307, 1970.

[25] G.-J. Nam, C. J. Alpert, P. Villarrubia, B. Winter, and M. Yildiz., "ISPD 2005 placement contest benchmark suite," in *Proc. ISPD*, pp. 216–220, 2005.

[26] 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 *ACM International Symposium on Physical Design*, pp. 141–146, 2011.