A Generalized Methodology for Well Island Generation and Well-Tap Insertion in Analog/Mixed-Signal Layouts

RAMPRASATH S, University of Minnesota, USA
MEGHNA MADHUSUDAN, University of Minnesota, USA
ARVIND K. SHARMA, University of Minnesota, USA
JITESH POOJARY, University of Minnesota, USA
SONER YALDIZ, Intel Labs, USA
RAMESH HARJANI, University of Minnesota, USA
STEVEN M. BURNS, Intel Labs, USA
SACHIN S. SAPATNEKAR, University of Minnesota, USA

Well island generation and well tap placement is an important problem in analog/mixed-signal (AMS) circuits. Well taps can only prevent latchups within a certain radius of influence within a well island, and hence must be appropriately inserted to cover all devices. However, existing automated AMS layout paradigms typically defer the insertion of well taps and creation of well islands to a post-processing step after placement. This alters the placement, resulting in increased area and wire length, as well as circuit performance degradation. Therefore, there is a strong need for a solution that generates well islands and inserts well taps *during placement* so that the placer can account for well overheads in optimizing placement metrics. In this work, we propose a modular solution using a graph-based optimization scheme that can be used within multiple placement paradigms with minimal intrusion. We demonstrate the integration of this scheme into stochastic, analytical and designer-driven row-based placement. The method is demonstrated in advanced FinFET technologies. Layouts generated using this scheme show better area, wire length, and performance metrics at the cost of a marginal runtime degradation when compared to the post-processing approach. Using our scheme, there is an average improvement of 3% and 4% and a maximum improvement of 23% and 11% in area and wirelength respectively of layouts of various classes of AMS circuits at the cost of 17% average and 29% maximum increase in total runtime.

Additional Key Words and Phrases: Well island generation, Well tap sharing, Analog circuits, Mixed-signal circuits, AMS layout automation, Circle graph, Jordan curve

ACM Reference Format:
Ramprasath S, Meghna Madhusudan, Arvind K. Sharma, Jitesh Poojary, Soner Yaldiz, Ramesh Harjani, Steven M. Burns, and Sachin S. Sapatnekar. 2022. A Generalized Methodology for Well Island Generation and Well-Tap Insertion in Analog/Mixed-Signal Layouts. *ACM Trans. Des. Autom. Electron. Syst.* 1, 1 (November 2022), 24 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

This work is supported in part by the DARPA IDEA program, as part of the ALIGN project, under SPAWAR Contract N660011824048.

Authors' addresses: Ramprasath S, University of Minnesota, Minneapolis, MN, USA; Meghna Madhusudan, University of Minnesota, Minneapolis, MN, USA; Arvind K. Sharma, University of Minnesota, Minneapolis, MN, USA; Jitesh Poojary, University of Minnesota, Minneapolis, MN, USA; Soner Yaldiz, Intel Labs, Hillsboro, OR, USA; Ramesh Harjani, University of Minnesota, Minneapolis, MN, USA; Steven M. Burns, Intel Labs, Hillsboro, OR, USA; Sachin S. Sapatnekar, University of Minnesota, Minneapolis, MN, USA.

1084-4309/2022/11-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

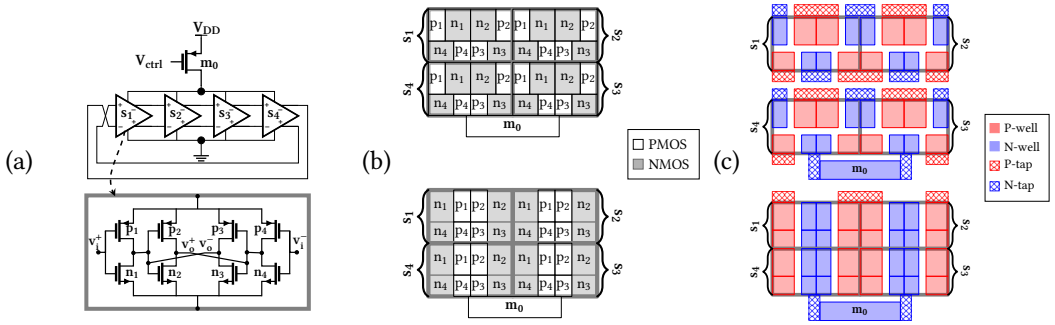


Fig. 1. (a) An inverter-coupled VCO; (b) Two post-placement layouts of the VCO; (c) The corresponding layouts of the VCO after well island definition and tap placement.

1 INTRODUCTION

Automated analog/mixed-signal (AMS) placers take a netlist/schematic, placement constraints, and transistor (MOSFET) level layouts as inputs and generate a *legal* placement as output that honors all the constraints. The transistors in a netlist could be one of many channel types (P, N, Deep N, or Deep P channel) and each channel type is associated with a corresponding well layer in the layout. This layer constitutes the bulk regions of the transistors. A continuous region of same well type forms a *well island*. Well islands must be connected to a corresponding power supply through a *well tap* cell that is typically provided by a foundry as part of the process design kit (PDK). These connections can prevent latchup [1, 2] within a well-layer-specific radius of influence (denoted $\mathcal{R}_w$ for well layer w) that is specified in the PDK. Different well type islands must be separated in the layout by a spacing specified in the PDK. AMS designs face stringent performance constraints that require low parasitics between critical transistors. This forces different well type transistors to be proximate in the layout, thereby creating complex well islands and tap placement problems.

Existing approaches to solve the well island/tap insertion problem in the literature have considerable scope for improvement since they either ignore or partly consider the impact of well overheads during placement. Moreover, the well island generation and well tap insertion problems are solved one after another even though they are intertwined. The detrimental effect of these well overheads on placement quality is illustrated using three state-of-the-art AMS placement paradigms: (a) stochastic [3–6], (b) analytical [7–9] and (c) manually-driven row-based placement [10, 11]. The stochastic and analytical placers use stochastic [12] and convex optimization [13, 14] algorithms respectively to generate an optimal placement by minimizing a cost function of layout area and wirelength estimates. In row-based placers, transistor layouts are restricted to a predefined height and are placed in rows similar to standard cells in a digital layout. The choice of devices and the order in which they are placed in each row is either extracted from the schematic [11] or specified by a layout designer [10]. Using layout examples, we will demonstrate that in each of these paradigms, well placement can impact the optimality of placement metrics such as area and wirelength.

In stochastic and analytical placement, well islands and taps are handled by a two-step post-processing scheme after placement: (i) well islands are generated by merging well regions of individual transistors using geometric operations [15] or generative adversarial networks (GANs) [16]; (ii) well taps are inserted by altering the placement. Although the GAN-based analytical placer proposed in [8], considers the impact of well islands and taps during global placement, it generates well islands using GAN and inserts well taps post placement. We will demonstrate the shortcomings of this post-processing approach using a differential-ring-oscillator-based voltage controlled oscillator (VCO) (Fig. 1(a)). Two post-placement layouts for the VCO are shown in Fig. 1(b), with four current-starved differential inverter stages, s_1 through s_4, of the VCO, each consisting of

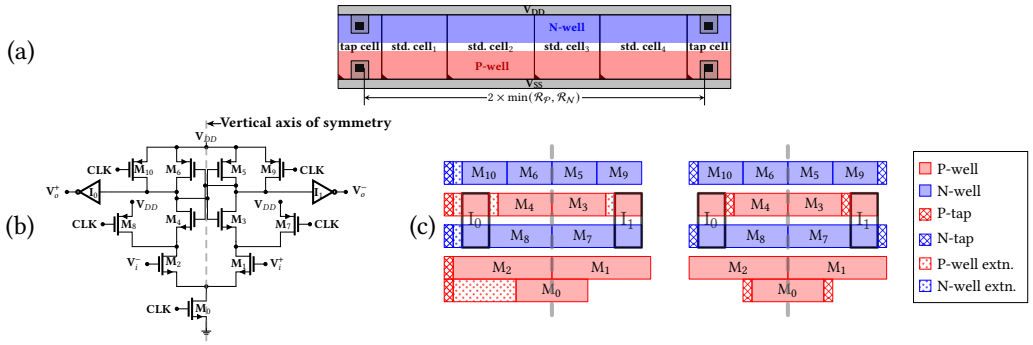


Fig. 2. (a) Digital layout with regularly spaced well taps; (b) An analog clocked comparator; (c) Designer-scripted row-based layouts of comparator with different well tap configurations.

four PMOS and four NMOS devices. These layouts have identical area and wirelength and hence are iso-cost prior to the well island/tap insertion. The corresponding post-processed layouts are shown in Fig. 1(c), where the P-well and N-well islands are in the red and blue colors, respectively; the islands are separated by PDK-specified well-to-well spacing, and well taps are shown by the crosshatched regions. The layout on the top has 35% larger area than that at the bottom, which leads to larger parasitics and degraded performance. The inability of the placer to distinguish between these layouts arises from the non-consideration of well overheads during placement. As far as we know, there is only one published technique that handles well islands during stochastic placement [17], and it only considers restricted island geometries and ignores the impact of well taps.

Next, let us look at the row-based placement approach. A surge in the number of design rules and layout dependent effects [18] in advanced FinFET technologies has led to the exploration of row-based placement for AMS layouts [10, 11]. We consider row-based placement approaches in which the relative positions of the transistors are specified by designer using scripts [10] or through the schematic [11]. In this approach, every row corresponds to a single well layer and hence can only accommodate transistors of that well type.[1] Every row is a well island and requires one or more well taps. Although this row-based placement is similar to digital placement, the digital well tap solution could be ill-suited for AMS layouts: the well taps are systematically inserted at regular intervals, determined by the tap radius of influence, as illustrated in Fig. 2(a). Consider the clocked comparator in Fig. 2(b). The comparator schematic is completely symmetric with respect to the vertical axis (dashed line in Fig. 2(b)); the left half of the schematic is a mirrored replica of the right half. This circuit operates on principles different from the VCO in Fig. 1. Therefore, unlike the VCO, which has no symmetry requirements, this layout is expected to have such symmetric positions for various transistors, well islands, and well taps for good performance matching between the two halves. Two layouts of the comparator are shown in Fig. 2(c) and the expected line of symmetry is shown using the dashed lines. The left layout uses the digital tap approach, and the right layout has the optimal taps. Unlike the digital placement, each row in an AMS placement need not be completely occupied by transistors. Hence the well layers must be extended to create contiguous well islands. The dotted regions in the left layout represent such well extensions, and it can be seen that it has a larger area and asymmetric well taps compared to the right layout. An algorithm

[1]Note that the layout in Fig. 1(c) contains different well type transistors in same row, and the corresponding layout is therefore not row-based.

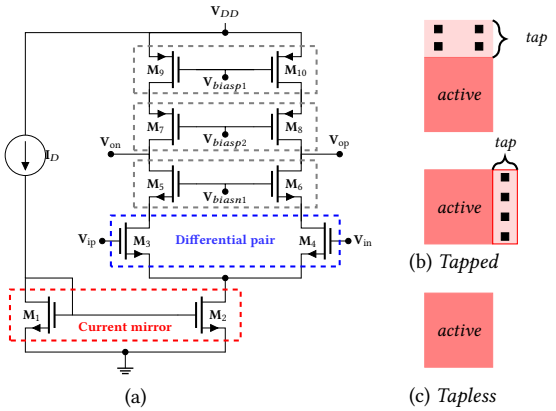


Fig. 3. (a) Telescopic OTA without a bias circuit; (b) and (c) NMOS differential pair (M_3,M_4) layouts.

that considers the impact of well taps on wirelength, area and placement constraints is required to generate the layout on the right.

This paper proposes a generic framework to optimally generate well islands and insert well taps for AMS layouts using a novel graph-based algorithm that can easily be integrated into multiple state-of-the-art AMS placement techniques. Section 2 covers preliminaries, graph-based constructs, and the modifications to the inputs of placers required to support this framework. Next, Section 3 describes the algorithms for well island generation and for finding the optimal number of well taps. Section 4 briefly describes the AMS placement techniques and the steps to integrate the proposed algorithm into each technique. Section 5 demonstrates the improvements in placement quality achieved by our algorithm, and finally, Section 6 concludes the paper.

2 WELL TAP GRAPH FORMULATION

2.1 Preliminaries

We use the telescopic operational transconductance amplifier (OTA) in Fig. 3(a) as a running example in this section. In our approach, we assume that library-based annotation [19, 20] is first used to identify primitive subcircuits in the netlist (using the terminology of [20]): the identified primitives of the OTA are labeled in Fig. 3(a). For each primitive, a cell generator constructs candidate DRC-clean layouts with various aspect ratios. This way of generating primitive layouts is common to all the current state-of-the-art placers though the technique used to identify the primitives varies between them.

For each generated primitive layout, the generator is modified to provide variants: *tapped*, with well taps, and *tapless*, without any well tap. A well tap may be vertical or horizontal or both, and different devices in the same circuit may have different tap orientations and have multiple taps. Fig. 3(b) and (c) illustrates two variants for the differential pair (M_3, M_4) in Fig. 3(a). Each tapped cell has an *active* and *tap* region, as shown in the figure. The input to our algorithm is a layout that comprised only tapped primitives. This is guaranteed by ensuring that all the cells provided to the placer are all tapped versions of the primitives.

A well tap for well layer w has a radius of influence, $\mathcal{R}_w$, defined in the PDK. The bulk impedance between the tap and any active device within this distance lies within the foundry-characterized limit for overcoming latchup. The locus of regions of equal substrate resistance from a point is a Euclidean circle. Typically, $w \in \{\mathcal{N}, \mathcal{P}, \text{Deep } \mathcal{N}, \text{Deep } \mathcal{P}\}$, representing an N-well, P-well, Deep N-well and Deep P-well, respectively; each has a different $\mathcal{R}_w$. To ensure that the depletion region

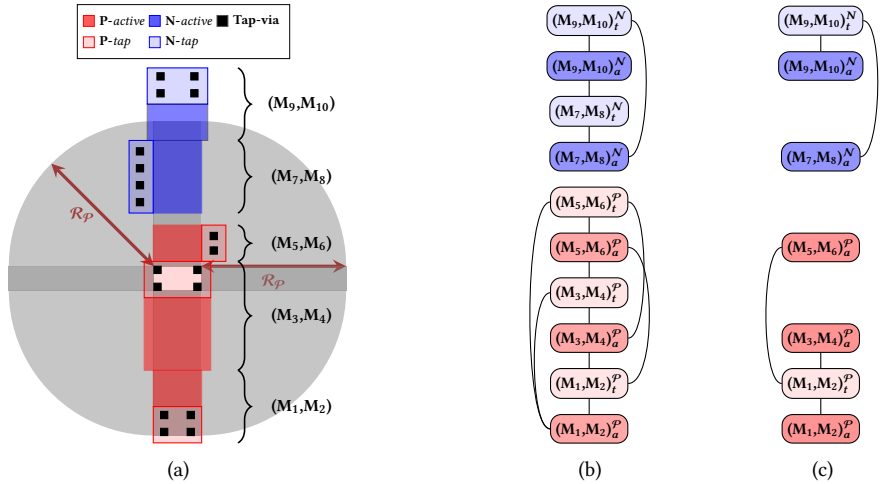


Fig. 4. (a) Layout of telescopic OTA in Fig. 3(a). (b) Its equivalent well tap graph. (c) The optimal tap vertices.

of different device types do not overlap during operation, well islands of different well types must be separated in the layout by a minimum spacing specified in the PDK. The input layout to our algorithm is assumed to honor this minimum spacing constraint between cells of different well types. Mainstream placement paradigms can easily be modified to support this requirement, as will be shown in Section 4.

2.2 Graph Construction

We devise a formulation to generate a layout with optimal well taps and well islands. The input is a placement P, specified using coordinates of all the cells and their orientation, where all cells are *tapped*. Our formulation detects and replaces redundant *tapped* cells in P with their *tapless* version, and then compacts the layout.

For the core problem of detecting redundant taps, we construct a undirected well tap graph $G(P) = (V, E)$. The vertex set $V = A \cup T$, where the vertices in A and T represent the set of *active* or *tap* cells, respectively. An *active* vertex C_a is connected to a *tap* vertex D_t by an undirected edge if (a) C_a lies completely within distance $\mathcal{R}_w$ of D_t, and (b) C_a and D_t belong to the same well island. The graph G is bipartite with A and T forming the two parts.

Fig. 4(a) shows a candidate layout of the telescopic OTA, where each cell X has both *active* (X_a^w) and *tap* (X_t^w) regions, represented by darker and lighter red/blue rectangles, respectively. The continuous P-well region comprising the active and tap regions of (M_1,M_2), (M_3,M_4) and (M_5,M_6) forms a P-well island, while the remaining devices form an N-well island. The vertical space between the cells (M_5,M_6) and (M_7,M_8) is required to honor the spacing requirement between the N-well and P-well islands.

The halo region around the tap for (M_3,M_4), shown by the gray region, completely envelopes the active regions of cells (M_1,M_2), (M_3,M_4) and (M_5,M_6); therefore, it is a valid tap for these cells. Although the halo overlaps the PMOS devices (M_7,M_8), it is not a valid tap for these as they lie in a different well. Fig. 4(b) shows the corresponding well tap graph G with the edges between active and tap vertices. Here, $(M_3,M_4)_t^\mathcal{P}$ has edges with each of $(M_1,M_2)_a^\mathcal{P}$, $(M_3,M_4)_a^\mathcal{P}$ and $(M_5,M_6)_a^\mathcal{P}$, the NMOS devices that lie within its halo. The number of strongly connected components of the graph G correspond to the number of wells: here, the two components correspond to the N-well vertices (shown in shades of blue) and P-well vertices (shown in shades of red).

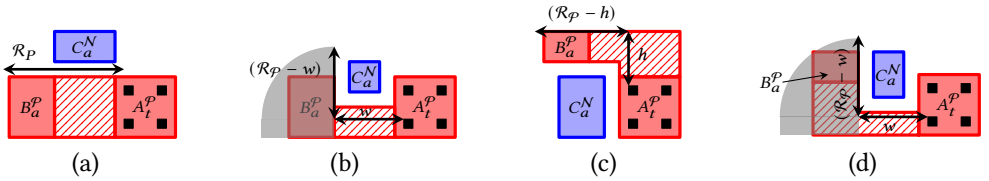


Fig. 5. (a) Unobstructed and (b) partially obstructed straight-line routes between two cells of same well type. L-shaped route between two vertices of same well layer: (c) no line-of-sight (d) fully obstructed line-of-sight.

3 DETAILS OF THE ALGORITHM

We now describe the key steps of the algorithm. For a given placement P, we first define a set of well islands, mapping the problem to that of planar routing. Next, we determine the optimal choice of well taps for P based on an ILP formulation.

3.1 Well Island Definition

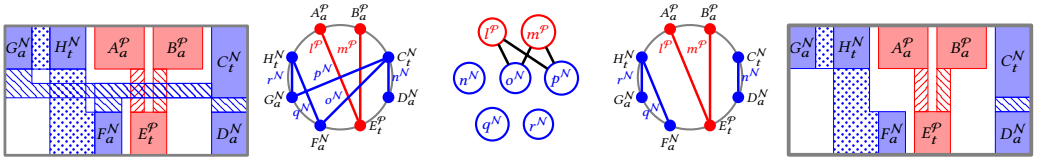
Based on the graph defined above, we first identify groups of active and tap cells that can share a *well island*. Each such well island must have at least one tap cell. Therefore, for each tap cell, we first identify the candidate active devices that can share its well island. A set of active cells can belong to the same well island as a tap cell if they are either placed adjacent to the tap cell, or can be connected to the tap cell in the layout with a rectilinear well layer shape. This connection is analogous to routing a net, whose pins are the active and tap cells, on a single layer (corresponding to the substrate). The active and tap cells of different well types thus become obstacles while routing this net. This has similarities to the problem of multipin single-layer routing, which is an NP-complete problem [21], of N-well and P-well pins. However, this problem is more complex, because it must also partition the set of N-well and P-well pins to optimize the overheads of well formation, as illustrated in Fig. 1(c).

To make the problem tractable, we solve a simpler problem of connecting two-pin connections between tap cells and active cells of the same well layer. We consider two types of routes:

- *Straight-line routes:* Two cells (pins) of the same well type can be connected using a rectangle-shaped route if they are in either vertical or horizontal line-of-sight and they are unobstructed or partially obstructed by a different well layer obstacle. Fig. 5(a) and (b) illustrate the unobstructed and partially obstructed scenarios with cells A_t and B_a of well type $\mathcal{P}$ and a cell C_a of well type $\mathcal{N}$ acting as an obstacle. The hatched rectangle shape in each of the figures represents the straight-line route.
- *L-shaped routes:* If two cells (pins) have no vertical or horizontal line-of-sight, or if the line-of-sight is fully obstructed by an obstacle, then the vertices could be connected using L-shaped routes. Fig. 5(c) and (d) illustrate L-shaped routes for the no-line-of-sight and fully-obstructed-line-of-sight cases, respectively.

The straight-line and L-shape routes are found using straightforward geometric operations, based on the coordinates of the cells to be connected by the route, and those of intermediate obstacles.

Two routes (well islands) of same well type can overlap to create a larger island. The well separation makes it *illegal* for two well islands of different well types to be within a minimum well separation of d_w. Two well islands that have a separation less than d_w represent an illegal *short* between the corresponding two nets. A legal island configuration is a collection of islands that have no illegal shorts between any two well islands. To obey design rules, the pins of the nets and the routes for dissimilar well types must be spaced apart by d_w. This is enforced by adding well separation constraint between the cells of different well types during placement.



(a) (b) (c) (d) (e)

Fig. 6. (a) Layout with overlapping well islands; (b) Jordan curve corresponding to layout in (a); (c) Circle graph corresponding to the curve in (b); (d) Optimal nonoverlapping edges; (e) Optimal non-overlapping well islands.

As an example of illegal island formation, consider the configuration in Fig. 6(a) with eight pins, $\{A_a^{\mathcal{P}}, B_a^{\mathcal{P}}, C_t^{\mathcal{N}}, D_a^{\mathcal{N}}, E_t^{\mathcal{P}}, F_a^{\mathcal{N}}, G_a^{\mathcal{N}}, H_t^{\mathcal{N}}\}$, three of which are tap cells. The pins are connected by a set of nets corresponding to the straight-line and L-shaped routes between active cells and tap cells $\{l^{\mathcal{P}} = (A_a^{\mathcal{P}}, E_t^{\mathcal{P}})$, $m^{\mathcal{P}} = (B_a^{\mathcal{P}}, E_t^{\mathcal{P}})$, $n^{\mathcal{N}} = (D_a^{\mathcal{N}}, C_t^{\mathcal{N}})$, $o^{\mathcal{N}} = (F_a^{\mathcal{N}}, C_t^{\mathcal{N}})$, $p^{\mathcal{N}} = (G_a^{\mathcal{N}}, C_t^{\mathcal{N}})$, $q^{\mathcal{N}} = (F_a^{\mathcal{N}}, H_t^{\mathcal{N}})\}$, $r^{\mathcal{N}} = (G_a^{\mathcal{N}}, H_t^{\mathcal{N}})\}$. The figure shows that this configuration is illegal due to shorts between the net pairs $(l^{\mathcal{P}}, o^{\mathcal{N}})$, $(l^{\mathcal{P}}, p^{\mathcal{N}})$, $(m^{\mathcal{P}}, o^{\mathcal{N}})$ and $(m^{\mathcal{P}}, p^{\mathcal{N}})$. Note that the overlapping net pair $(p^{\mathcal{N}}, q^{\mathcal{N}})$ is not a short as both nets are of the same well type.

To minimize the number of well islands while ensuring legality, a maximal number of nets must be routed without shorts. This is the maximal planar topological routing problem for two-pin nets [21, 22], which can be solved optimally with the following three steps:

Step 1: *Jordan curve representation:* A Jordan curve is a closed non-self-intersecting curve in a plane that divides the plane into two regions: the *interior* and the *exterior*. In our formulation, the Jordan curve is the smallest bounding box enclosing all pins under consideration. The interior (exterior) of the bounding box corresponds to the interior (exterior) of the Jordan curve. The pins that intersect the bounding box are placed on the perimeter of the curve either in a clockwise or counter-clockwise fashion. The pins contained inside the box are placed inside the curve. We will use the simplistic case of all pins being on the perimeter to first described the steps. Fig. 6(b) shows a Jordan curve (circle) equivalent of the layout in Fig. 6(a). The Jordan curve has all the pins on the perimeter placed clockwise. The chords $l^{\mathcal{P}}, m^{\mathcal{P}}, n^{\mathcal{N}}, o^{\mathcal{N}}, p^{\mathcal{N}}, q^{\mathcal{N}}$, and $r^{\mathcal{N}}$ represent the nets with straight-line and L-shaped routes. When two chords of different well types intersect in the circle graph, we have a short.

Step 2: *Circle graph construction:* A circle graph $CG = (V_c, E_c)$ is constructed for the Jordan curve, where V_c is the set of all the chords in the Jordan curve, and the edges in E_c represent the illegal shorts. Hence, two vertices in V_c are connected by an edge if the corresponding chords are of different well types and they intersect in the Jordan curve. Fig. 6(c) shows the circle graph corresponding to the Jordan curve in Fig. 6(b). The chords in Fig. 6(d) correspond to the nets. The circle graph has edges between nodes that correspond to the intersections of chords (nets) $l^{\mathcal{P}}$ and $m^{\mathcal{P}}$ with chords $o^{\mathcal{N}}$ and $p^{\mathcal{N}}$, which are shorted in Fig. 6(b).[2]

Step 3: *Maximum independent set (MIS) construction*: Supowit's algorithm [23] finds the MIS of circle graph CG, where MIS is the largest subset of V_c such that no two vertices in the subset are connected by an edge. The MIS of CG represents the maximum number of nets without shorts. Supowit's algorithm uses the fact that the MIS of a subgraph remains the same or increases by one element on addition of a new vertex to a subgraph. This fact is used to construct a dynamic programming approach that exhaustively searches through all possible subgraphs in $O(|V_c|^2)$ time, where $|V_c|$ is the cardinality of V_c. Fig. 6(d) shows the maximum number of non-overlapping edges found using Supowit's algorithm on the circle graph in Fig. 6(c).

[2]Note that this circle graph is **different from the well tap graph** defined in Section 2.2 and is used to find minimum number of legal well islands.

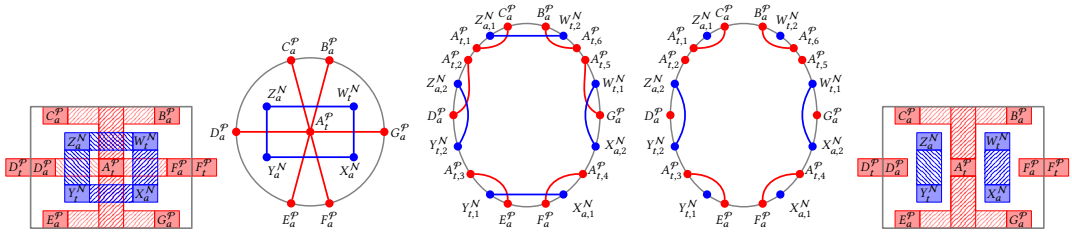


Fig. 7. (a) Layout with overlapping well islands; (b) Jordan curve corresponding to layout in (a) with all pins; (c) Transformed Jordan curve; (d) Optimal nonoverlapping edges; (e) Legal well islands.

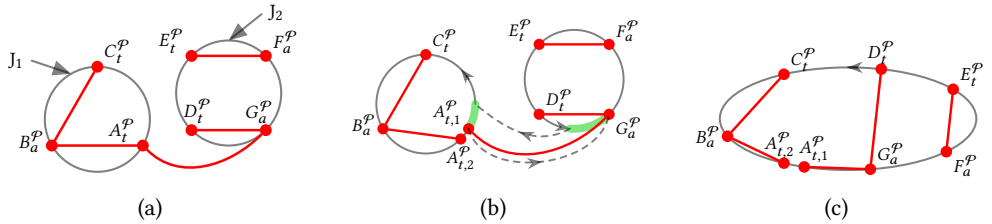


Fig. 8. (a) Smaller Jordan curves connected by a single edge; (b) Cut/stitched curves; (c) Merged Jordan curve.

After these three steps, we identify the minimal number of well islands: cells that are connected by a chord belong to the same island. In this example, the islands are $\{F_a^N, G_a^N, H_t^N\}$, $\{A_a^{\mathcal{P}}, B_a^{\mathcal{P}}, E_t^{\mathcal{P}}\}$, and $\{C_t^N, D_a^N\}$. Fig. 6(e) shows the layout without shorts corresponding to the chords shown in Fig. 6(d). The pins and edges that are not involved in any short can be ignored during the construction of the Jordan curve since they remain unaffected in all the three steps. For example, the edges (D_a^N, C_t^N), (F_a^N, H_t^N), (G_a^N, H_t^N) remain unchanged in Fig. 6(b), (c) and (d). This observation helps prune the circle graph and improve the runtime of the MIS algorithm.

In the example construction of the Jordan curve in Fig. 6(b), we only considered the scenario where all pins lie on the perimeter. We will now look at a more generic example layout with pins both on and inside the perimeter of the Jordan curve. Consider the layout in Fig. 7(a) with shorts between various edges connected to $A_t^{\mathcal{P}}$, W_t^N and Y_t^N. The Jordan curve created by the smallest bounding box of the layout is shown in Fig. 7(b). The pins $\{A_t^{\mathcal{P}}, W_t^N, X_a^N, Y_t^N, Z_a^N\}$ lie in the interior of the Jordan curve. A circle graph, by definition, uses chords of a closed curve as vertices and their intersection as edges. To construct a circle graph for this Jordan curve, the pins in the interior must be moved to the perimeter while maintaining the same set of shorts between edges. This transformation uses the following set of fundamental operations:

- The pins in a layout are partitioned into clusters, for each of which a *small* Jordan curve is constructed. Each cluster constitutes only of pins that can be placed on the perimeter of the small Jordan curves. Fig. 8(a) shows an example (different from the one in Fig. 7) with two clusters of pins: $\{A_t^{\mathcal{P}}, B_a^{\mathcal{P}}, C_t^{\mathcal{P}}\}$, and $\{D_t^{\mathcal{P}}, E_t^{\mathcal{P}}, F_a^{\mathcal{P}}, G_a^{\mathcal{P}}\}$ and their corresponding small Jordan curves, J_1 and J_2. The edges connecting the pins of a cluster lie within the corresponding small Jordan curve. The edges connecting pins on different clusters lie outside the curves. For example, the edge $(A_t^{\mathcal{P}}, G_a^{\mathcal{P}})$ lies outside both J_1 and J_2.
- The pins of multi-pin nets are split into as many *child pins* as the number of edges connecting them. All child pins belong to the same net as the original pin. For example, $A_t^{\mathcal{P}}$ in Fig. 8(a) will be split into two child pins $A_{t,1}^{\mathcal{P}}$ and $A_{t,2}^{\mathcal{P}}$ as shown in Fig. 8(b).
- When only a single edge connects two small curves, they can be *cut* and *stitched* to form a single Jordan curve. The cut and stitch operations ensure that (a) all edges, including the single

cross-connecting one, lie within the merged curve, (b) all pins lie on the perimeter of the stitched Jordan curve, and (c) the clockwise/counter-clockwise pin order is maintained after stitching. In Fig 8(b), the cut regions are shown in green and the stitches connecting J_1 and J_2 are shown using the dashed lines. The resultant merged Jordan curve is shown in Fig. 8(c), which has all pins on its perimeter in the same counter-clockwise order as in J_1 and J_2, and all edges within the perimeter.

- These operations are extensible to complex scenarios involving multiple edges between Jordan curves [22]. Successive pairwise merging of small curves results in a Jordan curve of the entire layout with all edges within and all pins on the perimeter.

The Jordan curves generated by these operations on the example layout in Fig. 7(a) are shown in Fig. 7(c). Any short between two edges in Fig. 7(b) has a corresponding short between edges connecting either the same pins or child pins in Fig. 7(c). This preservation of shorts is required to guarantee that the removal of an edge in the transformed curve removes the same set of shorts in the original curve as well. For example, the edge $(W_t^{\mathcal{N}}, X_a^{\mathcal{N}})$ intersects $(A_t^{\mathcal{P}}, G_a^{\mathcal{P}})$ in Fig. 7(b), which corresponds to the intersecting edges $(W_{t,1}^{\mathcal{N}}, X_{a,2}^{\mathcal{N}})$ and $(A_{t,5}^{\mathcal{P}}, G_a^{\mathcal{P}})$ in Fig. 7(c). This transformed Jordan curve can now be used to construct a circle graph, and Supowit's algorithm can be used to find the maximum non-overlapping edges on this circle graph, the result of which is shown in Fig. 7(d). The corresponding legal island configuration is shown in Fig. 7(e).

3.2 Well Tap Optimization

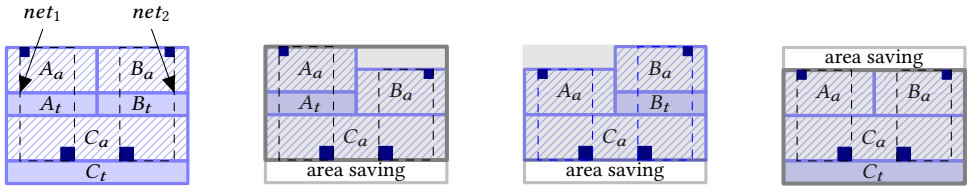
3.2.1 Preliminary Problem Formulation. Given the well tap graph $G(P)$ for a placement P, a tap vertex is said to **cover** an active vertex if there exists an edge between the two vertices. The task of well tap optimization is to find an *optimal* set S that covers all the active nodes and retain them. One simplistic definition of the optimal S is the set with least cardinality that covers all the active vertices. If a well tap node $t \in S' = T \setminus S$, it implies that all the active nodes that are covered by t are covered by one or more vertices in S. This indicates that the well taps corresponding to the vertices in S' are redundant and can be removed from the layout, i.e., the associated tapped cell is made tapless. Since S has the smallest cardinality, removing well taps in S' would superficially appear to yield an optimal layout. However, this optimization involves further subtleties, and as we will show below, removing all elements in S' does not necessarily provide an optimal layout.

3.2.2 Area and HPWL. To recognize the sources of suboptimality arising from using S with least cardinality, we must first understand the cost function used in analog layout synthesis. Typical analog placers [4, 5, 24–26] generate an optimal layout by minimizing the area and the estimated total wire-length for placement P. In our formulation, we consider the following cost function:

$$F(P) = \lambda_1 \cdot f_1\left(A\left(P\right)\right) + \lambda_2 \cdot f_2\left(L\left(P\right)\right) \tag{1}$$

where $A(P)$, the area of P, is the area of the smallest bounding box of all cells; $L(P)$ is the sum of the half-perimeter wire-lengths (HPWLs) of all the nets in the design; λ_i are the weights that indicate the relative importance of area and wire-length; and f_i are the normalization functions that map $A(P)$ and $L(P)$ to the range [0,1].

To show that the set S does not necessarily translate to maximal area and HPWL savings, which are the primary metrics used in (1), we consider the layout in Fig. 9 with three blocks, A, B, and C. The figure shows the bounding boxes of net_1 connecting A and C and net_2 connecting B and C. Fig. 9(a) shows the a layout where all blocks have well taps. If the radius $\mathcal{R}_w$ dictates that only one of these blocks is required to retain the well tap, then S could either be $\{A_t\}$, $\{B_t\}$ or $\{C_t\}$. Fig. 9(b), (c), and (d) show the compacted layout obtained on retaining each of candidate sets S. The area $A(P)$ of the layout is the same in Fig. 9 (b), (c), and (d), but the HPWL improvement depends on



(a) All taps (b) $S = \{A_t\}$ (c) $S = \{B_t\}$ (d) $S = \{C_t\}$

Fig. 9. Removal of well taps vs area/HPWL savings; Shaded region represents the bounding box used for area calculation; Dashed lines represent bounding boxes of nets net_1 and net_2 used for HPWL calculation.

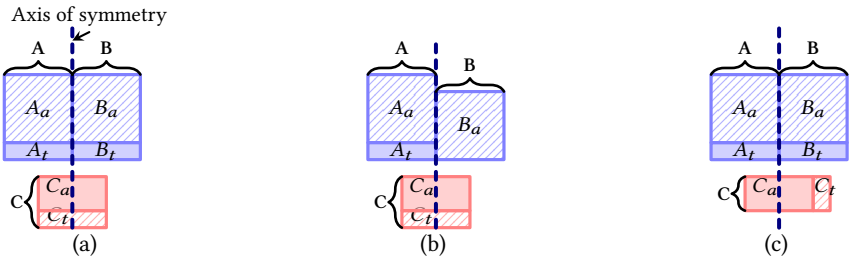


Fig. 10. Symmetry constraint honoring layout in (a); symmetry violation in (b) upon tap removal; self-symmetry constraint violation in (c).

the choice of S. It can be seen that the improvement in HPWL of net net_1 and net_2 for $S = \{C_t\}$ (Fig. 9(d)) is better than the other two cases. When A_t or B_t is retained, the HPWL of one net reduces while that of the other is unaltered, but when C_t is retained, the HPWL of both net_1 and net_2 is reduced. Thus, the impact of removal of well tap nodes on the area and HPWL must be considered to find an optimal S.

Fig. 9(b)–(d) also illustrate the factors to consider while calculating the area impact of tap node removal. The *void strip* created by removing C_t from Fig. 9(a) corresponds to the area saved in Fig. 9(b) and (c); the void strip created when A_t and B_t are removed together represents the area saved in Fig. 9(d). Thus, we save area if the removed taps align to create a void strip.

HPWL savings are possible when a tap node to be removed intersects the bounding box of a net and slices the box into two rectangles. For example, A_t intersects the bounding box of net_1 in Fig. 9(a) and bisects it into two rectangles, one each lying in A_a and C_a. The removal of A_t thus leads to reduction of HPWL of net_1, as seen in Fig. 9(c) and (d). Such an analysis is used to estimate the area and HPWL impact of removing well tap nodes.

3.2.3 Symmetry. A primary constraint used in analog circuit layouts is symmetry between blocks whose performance must be matched. *Self-symmetry* specifies line(s) of symmetry for a block such that its subhierarchies must be symmetric about the line(s). *Symmetry pairs* define pairs of blocks (typically of the same size) that must be placed symmetrically about a line of symmetry. A symmetry group is a collection of self symmetry and symmetry pair constraints that share a common axis of symmetry.

Fig. 10(a) shows an example layout satisfying a symmetry group with a vertical axis of symmetry, with a symmetry pair $\{A, B\}$ and self-symmetry for C. Upon removal of the tap B_t of cell B, A and B are no longer symmetrical as seen in Fig. 10(b). The asymmetry arises from removal of tap from one of the vertices in a symmetry pair. This implies that the two blocks belonging to a symmetry pair must simultaneously reject or retain the well taps in the layout.

(a) Well tap at corner (b) Well tap at center

Fig. 11. Centered well taps in (b) are preferred over (a).

For a self symmetric block, the tap vertex is forced to be symmetric in the input placement by forcing the placer to use the variant of the block with a horizontal (vertical) tap for a vertical (horizontal) axis of symmetry. This ensures that that the scenario in Fig. 10(c) does not occur. Thus, tap assignment must simply ensure that symmetry pair constraints are honored by retaining/removing the corresponding well taps simultaneously.

3.2.4 Centering. AMS designers prefer placing the well taps to be approximately equidistant from all the devices covered by the tap, so that the largest bulk impedance to any of the devices is minimized, making the tap maximally effective for all cells. Fig. 11(a) and (b) show two different valid well tap locations that satisfy the radius constraint: between these two variants, Fig. 11(b) is preferred since the largest distance from the well tap to any block is lower.

3.2.5 Refined Formulation. From the observations in Sections 3.2.2–3.2.4, we develop a new formulation for the optimal set S of well tap vertices that incorporates area and HPWL savings, symmetry constraints and encourages tap centering. We use an integer linear program that maximizes a weighted sum objective to identify S. The objective function of this ILP optimizes the area savings, HPWL savings, and tap centering, while the constraints enforce symmetry and ensure that all transistors are covered by a well tap.

If $x_i \in \{0, 1\}$ is the indicator variable corresponding to the presence of a tap node $i \in T$ in S, the objective and constraints are:

(1) Well tap coverage: Every active vertex in the graph G must be adjacent to at least one vertex in S to ensure that it has a well tap within distance $\mathcal{R}_w$. For each edge $e_{i,j} \in E$ between an active cell i and tap cell j, this is formulated as the ILP constraint:

$$\sum_{\forall (e_{i,j}) \in E} x_i \geq 1 \tag{2}$$

(2) Area and HPWL cost: The ILP formulation begins with a configuration where all cells have well taps and then determines the set of tap vertices to be removed. We can save area by only removing vertices that align in the layout horizontally or vertically, such that their removal creates a void strip. Let Q be the *set of sets* of all tap vertices whose well taps align, such that they create a void strip when removed: in Fig. 9(a), $Q = \{\{A_t, B_t\}, \{C_t\}\}$. Let h_k and w_k represent the height and width of a void strip formed by removal of $q_k \in Q$. If two different well type islands, one each on top and bottom (left and right) of a horizontal (vertical) void strip have a separation $sep < h_k + d_w$ ($sep < w_k + d_w$), then the removal of the void strip entirely would violate the well separation requirement. To honor well separation constraints and save area, instead of removing the entire horizontal or vertical void strip a slice can be removed from the strip, whose dimensions (w_{s_k}, h_{s_k}) are given by:

$$\{w_{s_k}, h_{s_k}\} = \begin{cases} \{w_k, \min\left(h_k, h_k + d_w - sep\right)\}, & \text{horizontal void} \\ \{\min\left(w_k, w_k + d_w - sep\right) h_k\}, & \text{vertical void} \end{cases} \tag{3}$$

The area saved by removing all tap vertices in set q_k, $\Delta Area_k = w_{s_k} \times h_{s_k}$. We use a single variable x_{q_k} to represent all tap cells in set q_k, so that they are all retained or removed together.

If the HPWL bounding boxes of N_i nets are sliced by tap node $i \in q_k$, the HPWL savings, ΔL_k, by removing all the taps in q_k are:

$$\Delta L_k = \left(\sum_{i \in q_k} N_i\right) \times \begin{cases} w_{s_k}, & \text{vertical void} \\ h_{s_k}, & \text{horizontal void} \end{cases} \tag{4}$$

The total area and HPWL savings upon removing all the vertices in $q_k \in Q$ can be expressed as:

$$x_{q_k} \times \left(\lambda_1 \Delta Area_k + \lambda_2 \Delta L_k\right) \tag{5}$$

where λ_1 (λ_2) is the importance of reducing area (HPWL), as in (1).
(3) Symmetry: If i and j are two tap vertices of two cells in a symmetry pair, then both tap nodes must be simultaneously retained or rejected by using the same ILP variable x_i for both tap cells.
(4) Centering: The *cover set* C_i of tap vertex i is the set of all active nodes that are adjacent to i in G. The range of the cover set, r_i, is the maximum Euclidean distance between i and its *cover set*. We augment the cost function to minimize r_i to incentivize the retention of tap vertices that are near the center of each cover set.

Combining all of the above, we formulate the following ILP:

$$\max_{x_i} \sum_{q_k \in Q} x_{q_k} \left(\lambda_1 \Delta Area_k + \lambda_2 \Delta L_k\right) - \sum_{i \in T} \lambda_3 x_i r_i \tag{6}$$

$$\text{such that} \sum_{\forall (e_{i,j}) \in E} x_i \geq 1, \begin{vmatrix} \forall j \in A \\ x_i \in \{0,1\}, \ \forall i \in T \end{vmatrix}$$

$$S = \{i | x_i = 1, i \in T\} \tag{7}$$

where λ_3 indicates the relative importance of tap centering. We obtain the set of retained well tap nodes from (6).

3.2.6 Computational Complexity. Our core ILP formulation, without pruning variables (e.g., using symmetry or void strip considerations) can be mapped to the *minimum weight dominating set* (MWDS) problem on a bipartite graph, an NP-complete problem [27].

Specifically, a set of vertices $D \subseteq V$ is a dominating set of G if every vertex in V is either in D or adjacent to one or more vertices in D. If every vertex $i \in V$ is associated with a weight w_i, then the minimum weight dominating set (MWDS) is a dominating set with smallest sum of weights. For example, for the layout of the telescopic OTA in Fig. 4(a), there are no symmetry or void strip constraints, S becomes a MWDS of G with the constraint that S is a subset of the set of tap cells, T, instead of the entire vertex set, V. The weight of vertex $i \in T$, w_i for this MWDS problem is:

$$w_i = (\lambda_1 \Delta Area_i + \lambda_2 \Delta L_i - \lambda_3 r_i) \tag{8}$$

Since MWDS is NP-complete, the complexity of its best-known optimal solution is exponential in $|V|$. We require $S \subseteq T$ instead of $S \subseteq V$, but this merely reduces the complexity to be exponential in $|T|$ instead of $|V|$. In practice, several factors ensure that the $|T|$ is not large, making the problem tractable: (1) void strip alignment constraints and symmetry pair constraints reduce $|T|$; (2) the use of hierarchy (Section 3.4) reduces the problem size at each hierarchical level. Finally, another mitigating factor is the structure of the problem: the dense connections in the well tap graph are localized within the Euclidean radius R_w of tap cells. This yields a sparse block-diagonal ILP constraint matrix. Typical ILP solution methods (e.g., branch-and-cut) solve an LP relaxation of the problem, and benefit from the sparsity and structure of this matrix.

(a) Before (b) After

Fig. 12. Routes being preserved after removal of redundant tap vertices; Optimal well taps $S = \{A_t, C_t\}$.

(a) Well tap at center (b) Well tap at boundary

Fig. 13. Suboptimality in hierarchical placement; H_2 hierarchy has two instances of H_1 one above another.

3.3 Well Island and Well Tap Layout

The goal of solving the ILP (6) is to identify well islands and well tap locations, so that the placement P corresponds to a legal solution. At this stage, we arrive at a placement that can retain only the tap vertices in S. We shift the locations of blocks in the placement to recover the area associated with unused tap cells, ensuring that well spacing requirements are met. If a tap cell of height h is deleted, its area is reclaimed by the layout by moving blocks above it downwards by distance h. A similar operation is used to move blocks to the left when a horizontal tap cell is deleted. The area and HPWL of the layout is estimated after this operation. These are useful in incremental cost evaluation when this technique is integrated into AMS placers.

After the area reclamation, the tap and active vertices must be connected to ensure all the required well islands are generated. The edges between vertices were guaranteed to be routable during the construction of G as described in Section 3.1. By construction, the reduction of redundant taps, which creates void strips that traverse the block, will ensure that the the updated layout remains routable. This is illustrated in Fig. 12(a) with three tapped cells A–C, where the optimal set $S = \{A_t, C_t\}$ and B_t can be removed. Fig. 12(b) shows the updated layout after removing B_t. In both figures, the hatched L-shaped region connecting C_t and B_a is routable, and the route is shortened after removing the redundant tap cell.

3.4 Hierarchical Placement

AMS designs are typically placed hierarchically, because design blocks tend to have a small number of components that can be logically grouped. Hierarchical placement can be visualized as a tree like structure, where each node represents a level of hierarchy and the children of the node are the blocks to be placed at that hierarchy level. The VCO in Fig. 1(a) has blocks s_1–s_4 at the first level of hierarchy, and inverter stages at the next level.

In fact, the use of hierarchy in placement is beneficial for the ILP formulation in (6), since a smaller problem is solved at every hierarchy. In practice, to avoid long runtimes, we impose a time limit for the solution of the ILP and choose the best solution obtained under that time limit. As a result, the ILP solution may not be optimal. Reducing the problem size increases the likelihood of obtaining an optimal layout at each level within the time limit. Placement at any hierarchical level is unaware of the global structure, and the locally optimum solution at some level of hierarchy may not be globally optimum for the overall layout. The well tap formulation inherits this problem for hierarchical placement: well taps that are optimal for the current hierarchy level may be suboptimal for the global layout.

We illustrate two scenarios in Fig. 13 using a layout that has two levels of hierarchy: H_1 and H_2, where H_1 has six cells (A-F) and H_2 has two instances of H_1 that are placed one above another, with mirror symmetry about the horizontal axis that separates them. In this case, the tap nodes E_t in Fig. 13(a) can be combined to retain a single well that covers all the devices in H_2, resulting in overall savings in the area and potentially HPWL. A second scenario is shown in Fig. 13(b), where no such union is possible. Thus, any tap nodes that are close to the boundary in a hierarchy may possibly combine with other tap node or can potentially cover active nodes in another hierarchy. Using this intuition, we add a distance term to the cost function in (6) that incentivizes the retention of wells that are closer to the boundary of a hierarchical block. If d_i is the Euclidean distance between node i and the boundary, the new ILP cost function, using another importance factor λ_4, is:

$$\max_{x_i} \sum_{q_k \in Q} \left(x_{q_k} \left(\lambda_1 \Delta Area_k + \lambda_2 \Delta L_i \right) \right) - \sum_{i \in T} x_i (\lambda_3 r_i - \lambda_4 d_i) \tag{9}$$

4 INTEGRATION INTO PLACERS

We will now look at the three broad classes of the state-of-the-art placement paradigms in the literature, namely: (a) stochastic, (b) analytical and (c) row-based placement and procedure to integrate our well island generation and well tap insertion algorithm into each of these paradigms. Initially, for all paradigms, the input cells to the placer comprise only the tapped cells as warranted in Section 2.

4.1 Stochastic placement

In this technique, a stochastic engine (typically simulated annealing) is used to generate the placement. This approach still remains the default choice for layouts with input primitives with multiple aspect ratios. The drawback of this approach is the long runtime required to achieve an optimal solution. A vanilla version of simulated annealing is described in Algorithm 1 [24], in which the cost $F(P)$ of placement P from (1) is optimized. In the algorithm, the triplet (T_{min}, T_{max}, α) forms the annealing schedule; *cooling parameter* α is a function of the size of the design and runtime limit for placement. A floorplan is first generated by the GenerateFloorplan routine that honors the placement constraints such as symmetry. The floorplan here is an encoding of the relative positions of the cells using schemes such as sequence pair [4], transitive closure graph [6], and B*-tree [3]. GeneratePlacement generates a compact placement from the encoding using either a constraint graph [24] or an ILP [5]. PerturbFloorplan perturbs the floorplan by either rearranging relative block positions or choosing a different aspect ratio for one or more cells and GeneratePlacement generates a new placement in successive iterations till the end of annealing schedule. The new placement is (a) always accepted if the cost reduces or (b) accepted with a probability that reduces with the temperature if the cost increases.

4.1.1 Integration of proposed scheme. The primary problem with the placer lies in the fact that it neglects well overheads during placement. To this end, we generate optimal well islands and well taps for the placement generated in each iteration of simulated annealing. This ensures the placer optimizes the *true* cost of the placement with all well overheads. The following steps are required to integrate our algorithm into the stochastic placer:

(1) Modification of GeneratePlacement: Our well island generation scheme expects the cells of different well types be spaced in the layout by the minimum well to well spacing d_w. This is achieved by adding this spacing requirement in the compaction method used by the GeneratePlacement routine in Algorithm 1. For the constraint graph approach, this involves the addition of constraint edges between cells of different well types. For the ILP approach, this

Algorithm 1 Placement using Simulated Annealing

```
1: function SimulatedAnnealing(N)          ▷ Annealing schedule: (T_min, T_max, α)
2:     T ← T_max
3:     f ← GenerateFloorplan(N)
4:     P_best ←GeneratePlacement(f, N)
5:     while T ≥ T_min do
6:         f ← PerturbFloorplan(f, N)
7:         P ←GeneratePlacement(f, N)
8:         if exp((F(P_best) − F(P))/T) ≥ random(0,1) then
9:             P_best ← P
10:        T ← T × α
11:    return P_best
```

involves adding a spacing inequality constraint between every pair of cells of different well types.

(2) Creation of well islands and insertion of taps: In every iteration of Algorithm 1, a well tap graph is constructed from P and the redundant well-taps are found using the ILP formulation in (6). The redundant taps are then removed by replacing the tapped cells with the corresponding tapless cells. P is then compacted to remove the whitespace using either the constraint graph or ILP whichever is used by the placer.

(3) Evaluation of the *true* cost: The cost of this updated placement is used to decide if a placement will be accepted or rejected. This helps the placer arrive at a true optimal solution.

4.2 Analytical placement

Analytical AMS placement [7–9] comprises two steps: (a) *Global placement* and (b) *Detailed placement.*

(a) Global placement: The placement problem is framed as convex optimization problem in which the cost function is a smooth convex approximation of area, wire-length, symmetry violation and cell overlap area. A convex optimization solver [13, 14] is used to obtain a minimum cost solution in this step. The cost function being optimized is typically of the form:

$$F(P) = \kappa_1 \cdot f_1(A(P)) + \kappa_2 \cdot f_2(L(P)) + \kappa_3 \cdot f_3(Sym(P)) + \kappa_4 \cdot f_4(\Theta(P)) \tag{10}$$

where A is the area, L is the wirelength estimate, Sym is the penalty function for symmetry constraint violation, Θ is the penalty function for overlapping area of cells, f_i are smoothing convex functions that approximate the respective quantities, and κ_i are scalars for relative importance of each component. The solution is typically illegal with overlapping cells and symmetry violations.

(b) Detailed placement: In this step, a legal placement is generated from the global placement. The relative positions of the cells are extracted from the solution and legalized using constraint graph [28], ILP [9], or a network-flow-based approach [7]. The analytical approach is significantly faster compared to the stochastic approach but has the disadvantage of not being able to choose between primitives that have multiple available aspect ratios during the global placement stage.

The integration of our proposed scheme into the analytical placement approach requires changes to both the global and detailed placement steps, as described below.

Global placement: To introduce the notion of well islands and well taps to the global placement step, we use the following intuition: (a) if the same well type primitives are proximate in the layout, it increases the likelihood of them belonging to the same well island; (b) a larger well island requires multiple well-taps. The reduced distance between same well-type cells helps reduce the sizes of

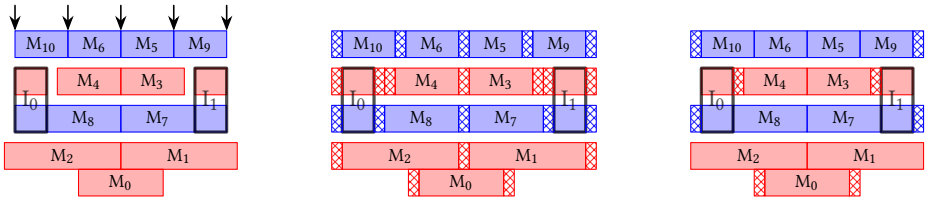


Fig. 14. Row-based layout of clocked comparator: (a) without well taps, (b) with well taps at all valid locations, and (c) optimal well taps.

the well islands which, in turn, reduces the number of well-taps. An extra term that captures the distance between same well type cells is added to the cost function [8]:

$$F(P) = \kappa_1 \cdot f_1(A(P)) + \kappa_2 \cdot f_2(L(P)) + \kappa_3 \cdot f_3(Sym(P)) + \kappa_4 \cdot f_4(\Theta(P)) + \kappa_5 \cdot f_2(W(P)) \quad (11)$$

where W is the sum of distances between same well-type cells.

Detailed placement: The detailed placement step is similar to the GeneratePlacement step of the stochastic placement approach. The changes (1) and (2) of Section 4.1.1 can be reused exactly for the detailed placement step. Well to well spacing for the different well type cells is inserted during detailed placement by adding appropriate constraints to the compaction routine. Once a detailed placement is generated with well-to-well spacing, a well tap graph can be constructed, well islands generated, and redundant well taps removed using our proposed approach. The layout is then compacted to remove the whitespace using the same compaction scheme used in detailed placement.

4.3 Row-based placement

In advanced FinFET process technologies, the large number of design rules and layout dependent effects greatly constrain the transistor layout geometries [18]. This has led to the exploration of a digital-placement-like approach for AMS circuits. The primitive layouts are restricted to discrete heights and are placed in rows similar to standard cells [10, 11], where every row comprises primitives of same well type. We describe the steps to integrate our approach into a designer-specified row-based placement scheme [11, 29]. These steps can be extended to other row-based placement approaches. In the scheme in [29], designer provides an abstract representation of the placement using the following specifications for each cell in the layout: (a) mirroring of the cell around x/y axis, (b) rotation of the cell in multiples of 90°, and (c) relative position of the cell {*right*, *left*, *top*, *bottom*} with respect to another cell in the layout. Each of these specifications is associated with a corresponding transformation matrix. The position of a cell is obtained by the transformation of position of its reference cell. The placement is thus a recursive application of these transformations starting with the cell at the origin of the layout. Another approach [11] obtains the relative positions of the cells from the schematic instead.

In this paradigm, the well island problem is solved by construction since each row comprises devices of a single well type, and is hence an island. The problem of finding the optimal well taps to be inserted into each island remains to be solved. As described in Section 1, the use of digital well tap solution for the row-based AMS layouts is suboptimal. Moreover, the layout designer is unaware of the exact position of the devices in the layout before the placement. This makes it hard

for the designer to specify relative locations of the well taps. This necessitates an automated well tap insertion approach.

The post-processing approach can be used to insert well taps but since it neglects the impact of wirelength and area during placement perturbation, it is an inapt choice. Our proposed well tap graph can be used here as demonstrated using the following example: Fig. 14(a) shows the layout of a comparator (Fig. 2(b)) generated using the row-based approach without well taps. For the top row, the potential locations of the well taps are indicated using arrows. If well taps were to be inserted at all these locations as illustrated in Fig. 14(b), our well tap graph scheme would identify the redundant ones to be removed. The automated well tap insertion approach using well tap graph for row-based placement involves the following steps:

- We first generate a layout using the row-based approach, with all the input cells having well taps on both left and right sides. We allow the well taps of adjacent cells in a row to either abut or completely overlap each other. This step generates the layout in Fig. 14(b) for the clocked comparator in Fig. 2(b).
- Next, we construct a well tap graph as described in Section 2.2. We can ignore the well island generation step in this construction since each row already is a well island.
- We then find redundant well taps using the ILP in (6). The impact of removal of a well tap on the overall wire length and area of the layout is considered in the ILP. This ensures that only the optimal taps remain in the layout.
- Finally, we compact the layout by calling the recursive transformation procedure used to generate the original placement.

Fig. 14(c) shows the layout with optimal well taps generated using this scheme. An important point to observe here is that symmetry of the layout is preserved after tap insertion.

5 RESULTS

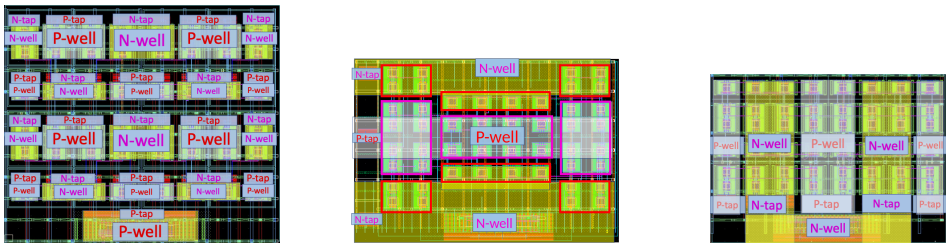
The well island generation and tap sharing optimization routine is implemented in C++ and compiled using GCC 8.2.0. The ILP solver lp_solve [30] is used to find the optimal number of taps using the formulation in (6). A time limit of 1ms is set for the solver to arrive at an optimal solution. Using the steps described in Section 4, this routine is integrated into three different placement algorithms: a) stochastic placement [20], (b) analytical placement [9], and (c) row-based placement [29]. Each of these placers is used to generate layouts for various AMS circuits using a commercial 12nm PDK on a Linux server with Intel Xeon(R) 2.20GHz Silver 4114 processors with 160GB memory. The tools Cadence Spectre, Calibre nmLVS and Calibre xACT are used for circuit simulation, layout vs schematic checking and parasitic extraction of the layouts respectively. The simulated performance of layouts generated by our proposed scheme is compared against other approaches in the following subsections, each of which corresponds to a different placement paradigm.

5.1 Stochastic placement

For the stochastic placement, we compare our approach, which generates well islands and optimal well-taps *during placement*, against:

Approach (A), which generates layouts using a layout generator based on [20], with built in well taps for each cell in the layout. These layouts honor $\mathcal{R}_w$ constraints by treating each cell as an island with its own tap and do not require a separate well island generation step.

Approach (B), which uses the placer [20] to generate layouts without any well taps, and then, *for that specific placement,* manually generates optimal well islands (minimum number of islands with optimal HPWL and area) and inserts well taps. This is the best achievable result from an approach (e.g., WellGAN [16]) that generates well islands and insert well taps *after* optimal well-oblivious



(a) Approaches (A) and (B) (b) Approach (C) (c) Proposed approach

Fig. 15. Comparison of layouts of differential-ring-oscillator-based VCO generated using various approaches.

placement.

Approach (C), which uses the placer [20] with an additional term in the objective that penalizes the distance between (a) same well type cells having similar dimensions or (b) different well type cells that have exactly same width and height or (c) cells which are different instances of same master device [17]. If i and j are pairs of cells that satisfy any one of the three conditions and $index_p(index_n)$ represents the positions of cells in the positive(negative) sequence pair [4], the penalty function $R(P)$ as described in [17] is given by:

$$R(P) = \kappa_3 \cdot \sum_{\forall i,j} \left(|index_p(i) - index_p(j)| + |index_n(i) - index_n(j)| \right)$$

The intuition behind the penalty function is that if similar dimension cells are subsequent in the sequence pair, then they are likely to be proximate in the placement. If the proximate cells are of similar dimensions and same well type then their well layers can merge to create rectangular well islands. If the cells are of different well type and are of same dimension then they can be placed in a regular rectangular array-like structure. In our implementation of this approach, well-taps are inserted manually post-placement at optimal locations.

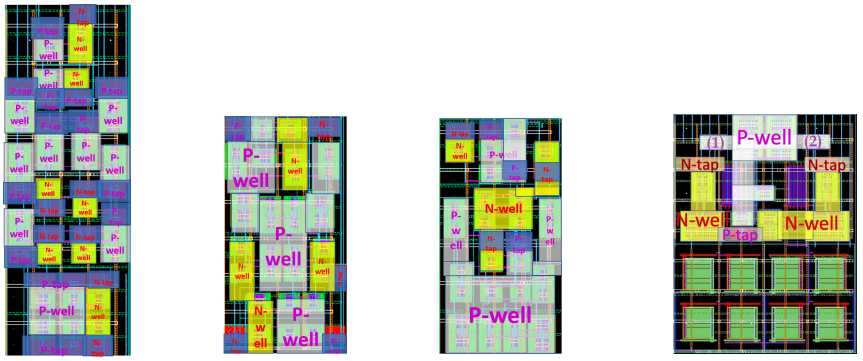
From Table 1, the area and HPWL for our method are much better than Approach (A), and sometimes noticeably superior to the manual Approach (B). in Approach (C) helped improve only the layout of the differential-ring-oscillator-based VCO, which has multiple cells of similar dimensions. There are no improvements in other circuits as seen in Table 1. In these circuits, either the cells are of vastly different dimensions or the cells with similar dimensions are already placed adjacent due to wirelength constraints. This implies that the penalty function has no additional impact on the placement. Post-layout performance metrics (Table 2) from our method are generally superior to Approaches (A), (B), and (C). For all tested layouts, the ILP solver is able to find an optimal solution in every iteration of the placement within the 1ms limit.

Fig. 15 compares the layout of the differential-ring-oscillator-based VCO in Fig.1(a), generated using Approach (A) and our proposed approach. Approach (B) generates a similar layout as Approach (A), and it is not separately shown here. We achieve 23% lower area and 11% lower HPWL than Approaches (A) and (B). This translates to a 49% improvement in the maximum frequency of the VCO. The maximum frequency depends on the parasitics between the transistors, which are reduced in our approach. The placement from Approach (A) and (B) results in a suboptimal layout (as in the figure at top in Fig. 1(b),(c)) since it does not consider the impact of well islands, their separation and tap location during placement. Both layouts have optimal power routing, and it is the improved signal routing that is the cause of performance enhancement.

Fig. 15(b) shows the layout generated using Approach (C) which only considers rectangular islands during placement. These islands generated during placement are highlighted in the figure using red/pink boxes. These islands are merged manually to create rectilinear islands to insert well

Table 1. Comparisons of Area and HPWL against Approaches (A), (B), and (C); ΔArea, ΔHPWL are the area and HPWL savings, respectively (larger the better); ΔT_{Placer} and ΔT_{Total} are the runtime overhead (smaller the better) in the placer and in the overall layout generation flow, respectively.

Circuit	Approach (A)				Approach (B)		Approach (C)		Proposed				ΔArea (%)			ΔHPWL(%)			ΔT_{placer}(%) vs. (A), (B), and (C)	ΔT_{total}(%) vs. (A), (B), and (C)
	Area (μm^2)	HPWL (μm)	T_{placer} (s)	T_{total} (s)	Area (μm^2)	HPWL (μm)	Area (μm^2)	HPWL (μm)	Area (μm^2)	HPWL (μm)	T_{placer} (s)	T_{total} (s)	vs. (A)	vs. (B)	vs. (C)	vs. (A)	vs. (B)	vs. (C)		
Eight-stage VCO	195.08	104.42	44	65	160.25	94.70	160.25	94.70	160.25	94.70	57	72	17.9	0.0	0.0	9.3	0.0	0.0	29	11
Differential-ring-oscillator-based VCO	393.45	167.95	90	141	393.45	167.95	358.65	162.63	301.00	149.35	137	173	23.5	23.5	16.1	11.1	11.1	8.2	52	23
Comparator	171.91	103.17	21	52	121.62	83.76	121.62	83.76	116.21	74.18	22	43	32.4	4.4	4.4	28.1	11.0	11.0	4	-17
High speed comparator	224.89	101.58	74	120	184.73	93.90	184.73	93.90	184.73	93.90	99	121	17.9	0.0	0.0	7.6	0.0	0.0	34	1
Five transistor high-frequency OTA	79.23	58.46	11	18	71.56	52.26	71.56	52.26	71.56	52.26	15	21	28.3	0.0	0.0	10.6	0.0	0.0	38	20
Cascode current mirror OTA	171.86	85.35	27	40	145.78	81.69	145.78	81.69	141.89	77.94	35	46	17.4	2.7	2.7	8.7	4.6	4.6	28	17
Folded cascode OTA	59.61	42.38	57	90	49.67	37.58	49.67	37.58	49.67	37.58	87	109	20.0	0.0	0.0	11.3	0.0	0.0	52	21
Two stage differential OTA	284.33	134.20	35	51	255.24	121.43	255.24	121.43	255.24	123.98	46	62	10.2	0.0	0.0	7.6	-2.1	-2.1	30	21
LDO error amplifier	169.25	154.89	31	40	145.15	150.18	145.15	150.18	145.15	150.18	42	51	14.2	0.0	0.0	3.0	0.0	0.0	35	29



(a) Approach (A) (b) Approaches (B) and (C) (c) Proposed (d) Two stage differential OTA

Fig. 16. (a)–(c) Comparator layouts from various approaches. (d) Non-rectangular islands built by our approach.

taps at optimal locations. As seen in the layout, restricting only to rectangular islands is clearly insufficient. Layout from our proposed approach in Fig. 15(c) takes rectilinear islands into account and achieves a 16% lower area and a 8% lower HPWL compared to Approach (C), which translates to a 20% improvement in maximum frequency of the VCO. The improvement in frequency is from the reduced parasitics between the transistors.

Fig. 16(a)–(c) compare our layout for a comparator against approaches (A), (B), and (C). We save 32% area and 28% HPWL over Approach (A). Relative to the manual Approach (B), we achieve slightly lower area and an 11% HPWL improvement, primarily in power routing. This HPWL reduction reduces routing congestion and leads to a small performance improvement.

The layout for a two stage differential OTA in Fig. 16(d) illustrates more complex well structures built using our approach. This layout has straight-lined well "routes" (left N-well island), L-shaped "routes" (right N-well island; P-well island). Symmetry constraints are maintained, e.g., when P-taps of symmetric devices (1), (2) are removed and their wells are merged with the larger P-well island.

Table 1 compares the improvement in area and HPWL achieved using our proposed approach. The results are demonstrated for various classes of analog and mixed signal designs: VCO, comparator, OTA and error amplifier. The ΔArea and ΔHPWL column show the difference in area and HPWL with respect to the approaches (A), (B), and (C). The area and HPWL savings using the proposed approach are apparent from the columns ΔArea and ΔHPWL of this table. The ILP formulation takes each candidate placement and incurs a runtime overhead in solving it to find the optimal well taps. The table shows both the absolute placer runtime, T_{Placer}, and the percentage difference, ΔT_{Placer}, between the placer runtime of our approach and Approach (A). The placer runtimes for approaches (A), (B), and (C) are similar. While our method involves a small increase in the runtime, the end result is a legal placement with well island formation, and well taps that obey the $\mathcal{R}_w$ tap constraints. The gain in area and HPWL over the baseline justify the increased runtime.

When we compare the total runtime, T_{total}, of the physical design flow, including routing (shown in Table 1), the percentage change in runtime, ΔT_{total} with respect to Approach (A) is more modest. In fact, the removal of the redundant well taps aids in reducing the obstacles for the routing and can actually improve the runtime for routing: for cases like comparator, this causes a net *reduction* in the total runtime. The overall runtime of approaches (B) and (C) are significantly larger (several hours, as against at most a few minutes for our automated method) due to the manual effort involved.

Table 2. Stochastic placer: Performance comparison of the layouts generated with and without using well islands and well taps

Circuit	Performance metric	Approach (A)	Approach (B) / (C)	Proposed approach
High-speed comparator	Evaluation delay (ps)	64	60	60
	Precharge delay (ps)	48	47	47
Comparator	Evaluation delay (ps)	192	153	151
	Precharge delay (ps)	149	119	116
Five-transistor high-frequency OTA	DC gain (dB)	22.13	22.13	22.13
	3-dB freq. (MHz)	335	343	343
	UGF (GHz)	4.08	4.18	4.18
Differential-ring-oscillator-based VCO	Max. freq. (GHz)	3.16	3.16 (3.92)	4.71
	Min. freq. (MHz)	311.90	311.90 (321.30)	327.13
	Voltage range (V)	[0,0.5]	[0,0.5]	[0,0.5]
Folded cascode OTA	DC gain (dB)	13.40	13.75	14.80
	3-dB freq(GHz)	2.39	2.68	2.64
	UGF(GHz)	7.60	7.77	8.05
Two-stage differential OTA	DC gain (dB)	44.4	45.0	44.7
	3-dB freq(MHz)	4.04	3.52	4.41
	UGF(MHz)	685	810	798

5.2 Analytical placement

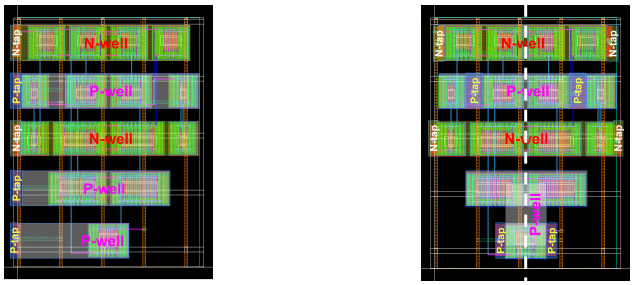
For the analytical placement approach, we compare our proposed well island generation and tap insertion routine against the same two approaches (A) and (B) used in the stochastic placement comparison in Section 5.1. Table 3 compares the dimensions and post-layout performance of three different circuits generated using analytical placement. Our proposed scheme simultaneously optimizes the layout for area, HPWL, well islands and the minimum number of well taps. This leads to the improved area and total wirelength of all the three circuits in Table 3. The improved wirelength reduces the parasitics and helps achieve better performance compared to the other two approaches. Since the entire well island generation and tap insertion procedure is called only once after the detailed placement step, our proposed scheme introduces insignificant runtime overhead. The runtimes to generate each layout is less than ten seconds using approach (A) and proposed scheme. As before, Approach (B) has a significantly larger runtime (hours) due to the manual effort involved.

5.3 Row-based placement

In the row-based placement approach, we compare the performance of layouts with well islands and taps generated using our scheme against the digital well-tap insertion approach. Fig. 17 shows the layout of clocked comparator (Fig. 2(a)) generated using the two approaches. The comparator is expected to have similar performance for positive and negative inputs. This imposes the symmetry requirement on the layout. Unlike the layout generated using digital well-tap approach in Fig. 17(a), the layout generated by our approach in Fig. 17(b) is symmetric around the vertical axis shown using the dashed line. Table 4 shows the difference in evaluation and precharge delays for the positive and negative inputs of the comparator for the two approaches. The asymmetry in digital approach layout leads to mismatch in performance between the two inputs of the comparator as seen in Table 4. This makes the layout generated by proposed approach preferred over the digital approach, even though they have same area and wirelength. In our approach, the ILP to find optimal well taps is invoked only once post placement, which adds insignificant runtime overhead to the total placement. Each of the layouts in Fig. 17 was generated within ten seconds.

Table 3. Comparison of dimensions and performance of layouts generated using analytical placement

Circuit	Performance metric	Approach (A)	Approach (B)	Proposed approach
Two-stage OpAmp	DC Gain (dB)	32.7	32.4	32.2
	3-dB freq.(MHz)	3.3	3.9	4.1
	UGF (MHz)	152	159	160
	Area (μm^2)	55.35	46.12	46.12
	HPWL (μm)	37.33	31.63	30.56
Telescopic OTA	DC Gain (dB)	11.2	11.1	11.2
	3-dB freq.(MHz)	800	840	847
	UGF (GHz)	2.8	2.8	2.9
	Area (μm^2)	67.73	58.31	55.31
	HPWL (μm)	56.17	52.20	50.99
Clocked Comparator	Evaluation delay (ps)	64	61	59
	Precharge delay (ps)	34	33	30
	Area (μm^2)	150.53	140.71	140.71
	HPWL (μm)	60.51	58.23	56.23



(a) Digital (b) Proposed

Fig. 17. Layouts of clocked comparator generated using digital and proposed approaches.

Table 4. Performance mismatch between positive and negative inputs of clocked comparator

Scheme	Positive input		Negative input		Area (μm^2)	HPWL (μm)
	Precharge delay(ps)	Evaluation delay(ps)	Precharge delay(ps)	Evaluation delay(ps)		
Digital	68	37	63	34	160.85	67.64
Proposed	63	33	63	33	160.85	67.64

6 CONCLUSION

A graph-based automated well island generation and well tap insertion algorithm for AMS circuits is proposed. The modular nature of this algorithm makes it easy to integrate it into existing state-of-the-art AMS placement techniques. This is demonstrated using three different placement paradigms: (a) stochastic, (b) analytical, and (c) row-based placement. The proposed algorithm helps estimate well overheads during placement, which enables the placer to optimize the true cost to achieve an optimal solution. Comparisons between the proposed algorithm and existing approaches using post-layout simulation of various classes of AMS circuits demonstrate the efficacy of this approach.

REFERENCES

[1] F. Farbiz and E. Rosenbaum, "Modeling and understanding of external latchup in CMOS technologies Part I: modeling latchup trigger current," *IEEE Transactions on Devices and Materials Reliability*, vol. 11, no. 3, 2011.

[2] R. R. Troutman, *Latchup in CMOS Technology: The Problem and its Cure*. Norwell, MA: Kluwer Academic Publishers, 1986.

[3] F. Balasa, "Modeling non-slicing floorplans with binary trees," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, 2000.

[4] F. Balasa, S. Maruvada, and K. Krishnamoorthy, "Using red-black interval trees in device-level analog placement with symmetry constraints," in *Proceedings of the Asia-South Pacific Design Automation Conference*, 2003.

[5] S. Koda, C. Kodama, and K. Fujiyoshi, "Linear Programming-Based Cell Placement With Symmetry Constraints for Analog IC Layout," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 4, pp. 659–668, 2007.

[6] B. Yang, Q. Dong, J. Li, and S. Nakatake, "Structured analog circuit design and MOS transistor decomposition for high accuracy applications," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, 2010.

[7] H.-C. Ou, K.-H. Tseng, J.-Y. Liu, I.-P. Wu, and Y.-W. Chang, "Layout-Dependent Effects-Aware Analytical Analog Placement," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 8, pp. 1243–1254, 2016.

[8] K. Zhu, H. Chen, M. Liu, X. Tang, W. Shi, N. Sun, and D. Z. Pan, "Generative-Adversarial-Network-Guided well-aware placement for analog circuits," in *Proceedings of the Asia-South Pacific Design Automation Conference*, 2022.

[9] Y. Lin, Y. Li, D. Fang, M. Madhusudan, S. S. Sapatnekar, R. Harjani, and J. Hu, "Are analytical techniques worthwhile for analog ic placement?" in *Proceedings of Design, Automation and Test in Europe*, 2022.

[10] E. Chang, J. Han, W. Bae, Z. Wang, N. Narevsky, B. NikoliC, and E. Alon, "BAG2: A process-portable framework for generator-based AMS circuit design," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, 2018.

[11] "Cadence Virtuoso: Doing Placement in a Row-Based Environment." [Online]. Available: https://community.cadence.com/cadence_blogs_8/b/cic/posts/virtuosity-doing-layout-in-a-row-based-environment

[12] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.220.4598.671

[13] R. Fletcher and C. M. Reeves, "Function minimization by conjugate gradients," *The Computer Journal*, vol. 7, no. 2, pp. 149–154, 1964.

[14] Y. E. Nesterov, "A method for solving the convex programming problem with convergence rate $O(1/k^2)$," *Doklady Akademii Nauk SSSR*, vol. 269, no. 3, pp. 543–547, 1983.

[15] J. Cohn, D. Garrod, R. Rutenbar, and L. Carley, "KOAN/ANAGRAM II: New tools for device-level analog placement and routing," *IEEE Journal of Solid-State Circuits*, vol. 26, no. 3, pp. 330–342, 1991.

[16] B. Xu, Y. Lin, X. Tang, S. Li, L. Shen, N. Sun, and D. Z. Pan, "WellGAN: Generative-adversarial-network-guided well generation for analog/mixed-signal circuit layout," in *Proceedings of the ACM/IEEE Design Automation Conference*, 2019.

[17] S. Nakatake, M. Kawakita, T. Ito, M. Kojima, M. Kojima, K. Izumi, and T. Habasaki, "Regularity-oriented analog placement with diffusion sharing and well island generation," in *Proceedings of the Asia-South Pacific Design Automation Conference*, 2010.

[18] A. L. S. Loke, D. Yang, T. T. Wee, J. L. Holland, P. Isakanian, K. Rim, S. Yang, J. S. Schneider, G. Nallapati, S. Dundigal, H. Lakdawala, B. Amelifard, C. Lee, B. McGovern, P. S. Holdaway, X. Kong, and B. M. Leary, "Analog/mixed-signal design challenges in 7-nm CMOS and beyond," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, 2018.

[19] T. Massier, H. Graeb, and U. Schlichtmann, "The Sizing Rules Method for CMOS and Bipolar Analog Integrated Circuit Synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 12, pp. 2209–2222, 2008.

[20] T. Dhar, K. Kunal, Y. Li, M. Madhusudan, J. Poojary, A. K. Sharma, W. Xu, S. M. Burns, R. Harjani, J. Hu *et al.*, "ALIGN: A system for automating analog layout," *IEEE Design & Test of Computers*, vol. 38, no. 2, 2020.

[21] A. Lim, V. Thanvantri, and S. Sahni, "Planar topological routing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, no. 6, pp. 651–656, 1997.

[22] M. Marek-Sadowska and T. T.-K. Tarng, "Single-layer routing for VLSI: analysis and algorithms," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 2, no. 4, pp. 246–259, 1983.

[23] K. Supowit, "Finding a maximum planar subset of a set of nets in a channel," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 6, no. 1, pp. 93–94, 1987.

[24] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "Rectangle-packing-based module placement," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, 1995.

[25] Y.-C. Tam, E. F. Y. Young, and C. Chu, "Analog placement with symmetry and other placement constraints," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, 2006.

[26] Q. Ma, L. Xiao, Y.-C. Tam, and E. F. Y. Young, "Simultaneous Handling of Symmetry, Common Centroid, and General Placement Constraints," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 1, pp. 85–95, 2011.

[27] F. V. Fomin, F. Grandoni, A. V. Pyatkin, and A. A. Stepanov, "Bounding the number of minimal dominating sets: a measure and conquer approach," in *Proceedings of the International Symposium on Algorithms and Computation*, 2005.

[28] K. Zhu, H. Chen, M. Liu, X. Tang, N. Sun, and D. Z. Pan, "Effective analog/mixed-signal circuit placement considering system signal flow," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, 2020.

[29] J. Han, W. Bae, E. Chang, Z. Wang, B. Nikolić, and E. Alon, "LAYGO: A Template-and-Grid-Based Layout Generation Engine for Advanced CMOS Technologies," *IEEE Transactions on Circuits and Systems I*, vol. 68, no. 3, pp. 1012–1022, 2021.

[30] M. Berkelaar, K. Eikland, and P. Notebaert, "lp_solve 5.5, Open source (mixed-integer) linear programming system," version 5.1.0.0 dated 1 May 2004. [Online]. Available: http://lpsolve.sourceforge.net/5.5/