Assessing Layout Density Benefits of Vertical Channel Devices

Wei-Che Wang, Charles Zhao and Puneet Gupta
Department of Electrical Engineering, University of California, Los Angeles

***Abstract*—Vertical channel devices have been considered as promising candidates for sub-5nm regime for the reduced area and large driving current. Several styles of layout designs and fabrication details of vertical channel devices have been proposed. However, due to the fast-changing manufacturing constraints for the advanced devices, the most efficient layout structures are still yet to be explored. In this paper we study the efficiency in terms of cell area of in-bound power vertical channel device layout, which is potentially the most compact vertical layout style. We develop and implement an efficient vertical layout generation framework for in-bound power layout to provide a quick evaluation of cell area given design rules and choices of folding strategies. The results are compared to vertically stacked lateral channel devices and out-bound power vertical channel devices. Both cell-level and chip-level comparisons show that in-bound power layout is more area-efficient than lateral devices and vertical out-bound power layouts.**

I. Introduction

As the demand of transistor scaling continues, conventional planar transistors or even FinFETs are beginning to face their limitations in the sub-5nm regime. Significant efforts have been devoted to the searching of alternative devices, and vertical devices have been considered as one of the most promising devices for future technologies [1, 2]. Many different vertical channel structures have been proposed in the past few years, such as vertical double gate [3] and vertical gate-all-around (VGAA) [4]. However, due to the fast-changing design rules and manufacturing constraints, layouts of vertical standard cells are constantly changing, making it difficult for layout designers to efficiently evaluate and design the smallest layouts given a new set of design rules.

A. Vertical Channel Devices

The main difference between a vertical channel device and a planar or FinFET device is that the channel of a vertical device is perpendicular to the wafer plane, so that the source and drain terminals of a vertical device can be aligned vertically to save cell area. Unlike vertical memory structure where the design is essentially arranging memory cells with same layout into arrays, the design of VGAA standard cells are much more complex given that each cell has its own schematic design, therefore an efficient design generation and evaluation framework for VGAA standard cell is needed. For planar CMOS, lateral gate-all-around (LGAA) [5], and FinFET, the layout generation methodologies have been studied, including search tree with lower bound pruning [6], or composition tree representations that guarantee the optimal transistor ordering [7]. The frameworks for an early stage design rule evaluation were also proposed [8, 9] for planer FETs. However, these algorithms cannot be applied to VGAA given that the structure of VGAA is radically different from planar or lateral FETs. In [10] a VGAA layout generation framework was proposed and implemented. However, the framework focuses on the out-bound power rail style, which may not be as area-efficient as the in-bound power rail style [11], where the VDD/GND rails are routed on top of VGAAs.

B. In-bound VGAA Standard Cell Layout Generation

The layout generation algorithm for the in-bound power rail VGAA is completely different from the algorithm for the out-bound VGAA presented in [10], in which the minimum edge covering algorithm tries to utilize as many vertical efficient structures as possible to minimize the number of diffusion gaps with large spacing rules. However, for in-bound VGAA, the power rails are routed in the middle of a cell, and intra-cell routing is done on the top and bottom of a cell, therefore such vertical efficient structures for out-bound VGAA do not exists for the in-bound VGAA, and a different layout generation algorithm for the in-bound VGAA is required.

In the layout generation framework, VGAA is considered symmetric, meaning that the direction of current flowing through the vertical channel can be either Source-On-Top (SOT) or Source-On-Bottom (SOB). Three routing layers are available: Metal1, Metal2, and bottom. Metal1 and Metal2 are on top of the VGAAs and they are unidirectional. Metal1 only routes along the Y-direction and Metal2 only routes the X-direction. The bottom layer is at the bottom side of VGAAs and its routing direction is not constrained.

The goal of in-bound VGAA layout generation is to find the layout structure with minimum number of tracks [12], which can be considered as a linear placement problem, where the best placement gives the least number of tracks required for each cell. Fig. 1 shows an example of how the placement of transistors can affect the number of tracks used. Fig. 1 (a) gives the schematic of the AOI standard cell. In Fig. 1 (b) the placement order is A, B1, and B2, where in Fig. 1 (c) the placement order is B1, A, and B2. Since B1 and B2 are parallel on the PMOS side, it is area-efficient to place them next to each other with both in SOT orientation to share the bottom layer without occupying one extra track as shown in Fig. 1 (c). Similarly on the NMOS side, it is area-efficient to place B1 and B2 next to each other to minimize the connection. The cell area in Fig. 1 (b) is smaller compared to Fig. 1 (c) with less number of tracks occupied. The goal of the layout generation is to obtain the placement order with the minimum number of tracks so that the area is minimized.

In this work we propose and implement a layout generation framework that takes design rules and different design styles, including the in-bound power rail, as inputs and generates compact standard cell layout correspondingly. By providing a quick evaluation of design constraints, the framework aims to facilitate the development of vertical advanced devices through effectively narrowing the gap between layout designers and manufacturers. Key contributions of this work are summarized as follows:

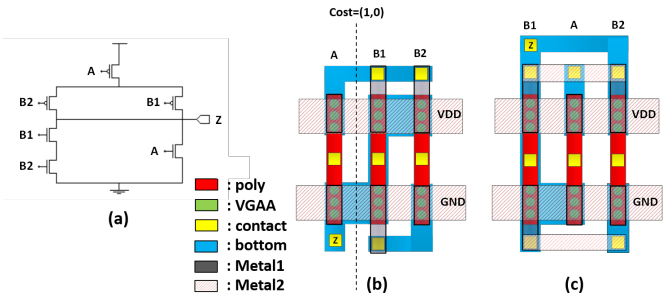


Fig. 1. In-Bound layout example. (a) AOI standard cell. (b) Placement with order A, B1, and B2. (c) Placement with order B1, A, and B2. It requires one extra track because B1 and B2 are not placed adjacently.

We propose and develop a VGAA layout generation framework incorporating the in-bound power rail design style, which is a more area-efficient layout style than the out-bound power rail design.

Cell-level and chip-level layout efficiencies of 4-stack LGAA, out-bound VGAA and variations of in-bound VGAA are compared to FinFET. Results show that VGAA has overall the best area efficiency, and in-bound VGAA is even smaller than out-bound VGAA.

II. In-Bound VGAA Layout Generation Algorithm

A. Cost Function

The cell area is obtained from the cell width and cell height, which can both be affected by the orientation (SOT or SOB) and placement of VGAAs as shown in Fig. 1. Therefore, finding the smallest area of a cell layout is equivalent to finding the thinnest and shortest layout placements.

The P_cost and N_cost between two transistors are defined as the implementation cost of the corresponding PMOS and NMOS structure, respectively. Finding the smallest cost is equivalent to finding the most compact cell implementation. The calculations of P_cost and N_cost include two parts: neighboring cost and global cost. Fig. 2 shows examples of neighboring cost and global cost.

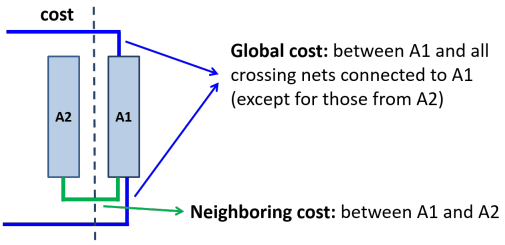


Fig. 2. Neighboring cost and global cost examples.

1) Neighboring Cost Calculation: The neighboring cost depends on the orientation and connections between the two VGAAs. Therefore, to calculate the neighboring cost, a Cost Look-Up-Table (C-LUT) is used. In C-LUT, the neighboring costs of all possible combinations between two adjacent VGAAs in a placement are provided as shown in Table I. Eight possible connections are listed with two VGAAs with stack or parallel structures and SOT and SOB orientations.

The corresponding connections of the C-LUT is given in Fig. 3. The cost of stack SOT+SOB is zero because the connection is done through the shared bottom layer without occupying a track. The cost of parallel SOT+SOT can be zero or one depending on the source of the VGAAs. If the source is the power rail, the track of Metal2 can be eliminated; otherwise, a track for the Metal2 connections is needed to connect the sources of the two VGAAs.

TABLE I
Cost Look-Up-Table. The cost of parallel SOT+SOT can be zero or one depending on the source of the parallel structure. If it is the power rail, the cost is zero; otherwise, the cost is one.

	SOT+SOT	SOB+SOT	SOB+SOB	SOT+SOB
Stack	1	1	1	0
Parallel	0 or 1	2	1	2

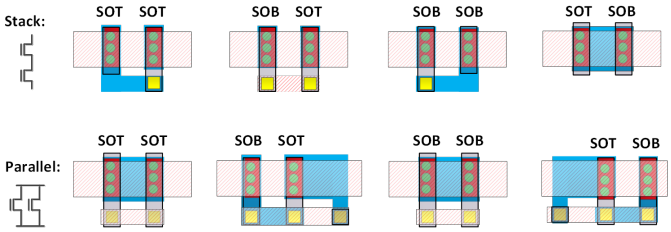


Fig. 3. Neighboring connections. NMOS type is used for illustration.

Please note that the connections of stack VGAAs shown in Fig. 3 are for two VGAAs with current flowing from the left VGAA to the right VGAA. For the cost of stack VGAAs with current flowing from the right VGAA to the left VGAA, the connections are mirrored horizontally with the same costs.

2) Global Cost Calculation: The calculation of global cost is much simpler than neighboring cost calculation because no bottom layer can be shared by two VGAAs in a global connection. Since a global connection can be formed using bottom layer or Metal2 layer, the global cost is 0.5 for each net. If the cost is 1.5, then two horizontal tracks would be needed for the global connection. When two global connections share a horizontal track, the inner one will be using bottom layer, and the outer one will be using Metal2 layer. Source-to-gate and drain-to-gate connections are treated as global connections because they require one connection to the top of polysilicon-gate terminals.

B. Placement Tree Representation

In the layout generation algorithm, the search space of linear placements of a cell is represented by a tree. Each node in the tree represents a (partial) linear placement of transistors except for the root node. Starting from the root node, which means no placement is done yet, the tree is constructed in such a way that the placement of a child node is the placement of its parent node with the next transistor placed at the right. Each node n contains a P_cost and a N_cost representing the costs between the placement of its parent node and the newly placed transistors on the right. Fig. 4 shows an example of a tree representation of the AOI standard cell placement. Node $n_{fA;B1g}$ contains the cost of the cut shown in Fig. 1 (b) where B1 is placed at the right side of A, and the number of tracks needed are one and zero on the PMOS and NMOS side, respectively. A leaf node of the tree represents a completed placement. The $depth$ of a node is defined as the number of edges from the node to the root node. The depth of a leaf node is the number of transistors, and all nodes with depth=1 have zero cost.

1) Path Cost Definition: Since the cell height is decided by the connection with the most tracks needed, for each node n, its path cost can be calculated, which is defined as

$$Path\_cost(n) = MAX\_P(n) + MAX\_N(n) + Adjustment(n) \quad (1)$$

where $MAX\_P(n)$ and $MAX\_N(n)$ are the maximum P_cost and N_cost of nodes on the path from the root node to n, respectively. $Adjustment(n)$ checks if n is a leaf node. If not, it returns zero; otherwise, the returned value depends on the exact connection of the path, which will be detailed in Section II-B2. In Fig. 4, the leaf node $n_{fA;B1;B2g}$ represents the placement shown in Fig. 1 (b), where the numbers of tracks between two transistors are shown in the corresponding nodes. The final placement cost is $1 + 1 + 0 = 2$.

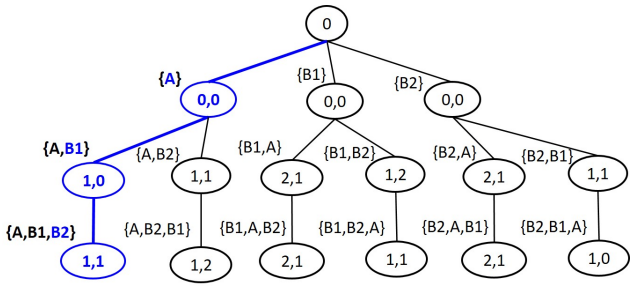


Fig. 4. Placement tree example. For illustration purpose all VGAAs are SOT in the example. The leaf node $n_{\{A,B1,B2\}}$ represents the smallest area placement as shown in Fig. 1 (b). Note that the search of minimum cost will not explore the entire tree.

2) Path Cost Adjustment: During the path cost calculation of a leaf, one adjustment needs to be done to reflect the final number of tracks needed to include output pins, for example pin Z in Fig. 1. An output net is defined as the net connected to an output pin, where an external connection to the net is required for output pin access. In a path from root to a leaf node, one of the nodes will be selected for the output pin and the cost adjustment will be done on the node. The node will be selected from the candidate nodes listed in the following groups:

1) Nodes that consist of global connections where the connections include output net.
2) Nodes with smallest P_cost or N_cost and the connections include output net.

Since an output pin needs to be connected externally, the goal of the adjustment is to find the connection location with the least impact to the path cost. For the candidates in the first group, since a global connection will require one track already, the output pin can be placed directly on the routing track without adding cost to the path cost as an example shown in Fig. 1 (c). If no candidates exist in the first group, a candidate from the second group will be selected. In such case, the node with the smallest P_cost or N_cost is selected and its P_cost or N_cost is incremented by one. As shown in Fig. 1 (b) and Fig. 4, the adjustment is added on the N_cost of node $n_{fA;B1g}$ which the cost becomes (1,1) after the adjustment. Since the path cost of $n_{fA;B1;B2g}$ is 2 before the adjustment, adding the N_cost to $n_{fA;B1g}$, which was (1,0), will not affect the path cost. If all nodes on a path have equal costs, it is possible that the path cost is incremented to accommodate the output pin.

From the tree we see that to implement the AOI standard cell, at least two tracks are needed for the intra-cell routing. Please note that the search of minimum cost will not explore the whole tree as the details of the branch and bound search algorithm will be provided in the following section.

C. Branch and Bound Search

The branch and bound search follows the Depth First Search (DFS) principle. Given a placement tree the procedure of the branch and bound search algorithm is described as follows:

1) An initial placement is created by a heuristic described in Section II-D1. The leaf node corresponding to the initial placement is set as the current minimum cost node n_{min}. The search bound is $Cost_{min} = Path\_cost(n_{min})$.
2) Going from the root node, a DFS is performed. The currently accessed node is defined as $n_{current}$ and its cost $Cost_{current}$ is compared with $Cost_{min}$.
3) If the condition $Cost_{current}$ $Cost_{min}$ is satisfied, $n_{current}$ and all its children nodes will not be further evaluated; otherwise, if $n_{current}$ is a leaf, minimum node and cost are updated as $n_{min} = n_{current}$ and $Cost_{min} = Path\_cost(n_{current})$.
4) The search stops when all remaining nodes are visited. The minimum number of tracks needed is $Cost_{min}$ and the placement is n_{min}.

D. Search Space Reduction Techniques

To reduce the search space, during the implementation, the placement tree is built dynamically as the branch and bound search is in progress. In addition, many techniques are applied to further reduce the search space, therefore the full branch and bound is run on all cells to guarantee an optimal result.

1) Initial Placement Heuristic: Since only those nodes with smaller cost than the minimum cost will be visited, the majority of the tree will be pruned if a good initial placement with small cost is used as the bound condition. It is clear to see in Fig. 3 that if two transistors are stacked, it would be efficient to put them next to each other with SOT+SOB to save one track. Similarly if two transistors are parallel and connected to power rail, it would be efficient to put them next to each other with SOT+SOT, and in general SOB+SOB would be more efficient than SOB+SOT or SOT+SOB for parallel transistors. From the observations described, an initial placement heuristic is implemented following these principles:

1) All transistors connected to power rails are SOT.
2) All transistors connected to outputs are SOT.
3) All transistors in a same paralleled structure on either NMOS or PMOS side are SOB+SOB or SOT+SOT and placed next to each other.
4) All transistors in a 2-stacked structure on either NMOS or PMOS side are SOT+SOB and placed next to each other.

These principles are not mandatory requirements but the initial placement heuristic will try to meet as many principles as possible to minimize the initial placement cost to effectively reduce the search space.

2) Mirrored Placement Removal: Two placements are mirrored when the orders of transistors are reversed. For example, a placement of transistor order A, B, and C is a mirrored placement of transistor order C, B, and A. Since two mirrored placements have the same number of tracks, only one of them needs to be evaluated. The removal of mirrored placement is equivalent to searching the left half of the tree only. As shown in Fig. 4, the leftmost path A,B1,B2 and the rightmost path

B2,B1,A are the mirrored placements, therefore only one of them should be evaluated.

E. Folding Strategy

One flexibility provided in the proposed layout generation framework is the selection of folding strategy, which specifies how many P-VGAAs and N-VGAAs a polysilicon shape can hold. For each VGAA, since the effective width is its diameter multiplied by [10], the number of VGAAs required for a cell depends on the driving capability of the cell, and how these VGAAs are accommodated can change the area of the cell. With more VGAAs on a polysilicon shape, the cell becomes taller but thinner; on the other hand, with less VGAAs on a polysilicon shape, the cell becomes shorter but wider. Fig. 5 shows the AOI cell implementation of 5 folding strategies examined in our experiments. The pair defines the number of P-VGAAs and N-VGAAs on a polysilicon shape as illustrated in the figure. In this example (3,2) is better than (2,1) because (3,2) needs 6 polysilicon shapes but (2,1) requires 10 polysilicon shapes. Please note that the generation algorithm is the same for all folding strategies, where a folding strategy can be transformed to another folding strategy by extending connections or merging polysilicons. Also, folding does not change the schematic of the original cell.

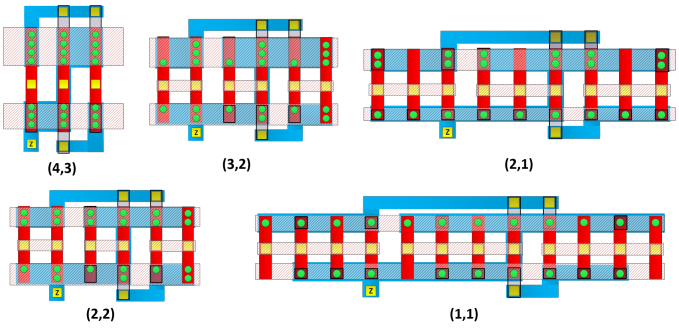
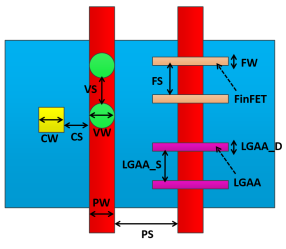


Fig. 5. AOI cell implementation with the 5 folding strategies compared in our experiments. The number pair indicates the number of P-VGAAs and N-VGAAs on a polysilicon. For example, (4,3) implies that a polysilicon can hold 4 P-VGAAs and 3 N-VGAAs.

III. Experimental Results

The proposed algorithm is implemented in C++ with the use of OpenAccess API [13]. A 28nm standard cell library, which contains no pass-gate or tri-gate structures, is scaled to a 7nm standard cell library for our experiments, and the design rules are given in Fig. 6 [12, 14]. For FinFET, the effective width of each fin is FW+Fin_Height 2 = 42, while the effective width of each LGAA and VGAA is LGAA_D 22 and VW 22, respectively. In our comparisons each polysilicon holds up to 5 FinFETs or non-stack LGAAs, and the number of VGAA on a poly is 5 because (3,2) is the best folding strategy as presented in Section III-A.

FinFET is used as the baseline in our comparisons. We first compare the 5 different in-bound VGAA folding strategies to FinFET, and in-bound VGAA with the best folding strategy is compared to vertically 4-stack LGAA [15] and out-bound contact reduction VGAA [10]. We present both cell-level and chip-level (after placement and route) comparison results since cell-level results can be misleading due to routing congestion [16]. The VGAA layout generation run time for the most



Design Rules	nm
CW	7
VS	20
VW	7
PW	7
PS	14
CS	11.5
FS	20
FW	6
Fin_Height	18
Fin_PS	21
LGAA_D	7
LGAA_S	20

Fig. 6. Design rules of the scaled 7nm technology for our experiments.

complex cells, for example a scan DFF, is only a few seconds for the effective search space reduction techniques presented in Section II-D.

A. Folding Strategy Comparison

Fig. 7 shows cell-level areas of folding strategies normalized to FinFET on 4 benchmarks [10]. In general, a better folding strategy is when the number of P-VGAAs on a polysilicon is larger than the number of N-VGAAs because the total width of PMOS is larger than NMOS for most cells for balanced rise and fall transition. (1,1) is not a good folding strategy because the number of polysilicon gates now becomes the number of VGAAs. (3,2) has the smallest area given the cells used in the benchmarks. Our framework allows layout designers to decide the best folding strategy given the cells used and design rules. Please note that the cell height of in-bound VGAA is the same for all cells and is determined by the cells with the highest cost, which are scan DFF cells with cost=5. In the following sections, the results of in-bound VGAA are based on (3,2) folding.

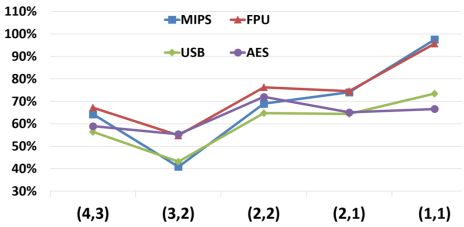


Fig. 7. Folding strategy comparison. (2,1) has the smallest area given the cells used in the benchmarks.

B. Single Cell Comparisons

Single cell comparisons to FinFET are presented in Table II. The negative values are the area reduction ratio, where the positive value indicates that the area is larger than FinFET. We can see that 4-stack LGAA shows area benefits for larger driving cells, but for small driving cells, since the required FinFETs can be fit in a single polysilicon shape already, vertically stacking LGAA does not contribute to area benefits. For out-bound VGAA and in-bound VGAA, much area reduction is observed for small driving cells, but for some large driving cells, the area reduction begin to diminish. For INV_X16 the in-bound VGAA is even larger than FinFET due to the large number of VGAAs needed. In general, in-bound VGAA is smaller than out-bound VGAA except for the large driving cell due to higher cell height.

C. In-bound and out-bound VGAA Comparison

The comparison results of cell-level and chip-level placement and route using a commercial tool [17] are presented