Detailed Placement for Leakage Reduction Using Systematic Through-Pitch Variation

Andrew B. Kahng[†‡], Swamy Muddu[‡], Puneet Sharma[‡]
[†]CSE and [‡]ECE Departments, University of California at San Diego
{abk, smuddu, sharma}@ucsd.edu

ABSTRACT

We present a novel detailed placement technique that accounts for systematic through-pitch variations to reduce leakage. Leakage depends nearly exponentially on linewidth (gate length), and even small variations in linewidth introduce large variability in leakage. A substantial fraction of linewidth variation is systematic with respect to the device layout context. Detailed placement changes context of the devices that are near the cell boundaries and can be used to reduce leakage. Our approach modifies the placement of cells in small windows such that contexts that reduce leakage are created. During this optimization, cells are partitioned into rows and then placed in rows using a traveling salesman problem formulation.

Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design Aids—*layout, placement and routing*

General Terms

Design, Power, Variability

Keywords

Lithography, ACLV, Through-pitch, Leakage, Detailed Placement

1. INTRODUCTION

Power, composed of dynamic and leakage power, is a primary design concern in modern designs. Leakage power has scaled worse than dynamic power with technology scaling, and its share of total power continues to increase. While gate-tunneling leakage can be large at low temperatures, subthreshold leakage is the dominant leakage component at operating temperatures. For 45*nm* and beyond, the use of high-K dielectrics is projected to significantly reduce gate-tunneling leakage. So, subthreshold leakage is likely to remain the dominant contributor to total leakage in foreseeable technologies. In this paper we focus on subthreshold leakage and refer to it as leakage. Leakage variability, the difference in leakage of different chips of the same design, is very significant and together with performance variation limits the yield. For 180nm technology, Intel reported a 20× leakage variation [4].

Device leakage depends nearly exponentially on polysilicon (poly) linewidth (gate length), and even small variations in linewidth cause large leakage variation. Across-chip linewidth variation (ACLV) has a substantial systematic component [17] which is predictable once the *pitch* and *defocus* of a device (line) are known [7, 9]. Pitch of a device captures the context of the gate of the device and in simple terms is the spacing of the gate from neighboring gates. Once placement has been performed, pitches of all devices in the design are known. Defocus is the variation in focus; it arises due to wafer topography variations, lens aberrations, tilt of the wafer stage, etc. and is difficult to predict. Recent works have used systematic ACLV for accurate timing analysis [9], design robustness [10], and leakage analysis and optimization [14]. Design rule manuals now present ACLV data to enable ACLV-driven optimizations and analyses.

Through-pitch variation is the linewidth variation that occurs over different permissible pitches. Reticle enhancement techniques (RETs), such as optical proximity correction (OPC) and scattering bar insertion, reduce but do not completely eliminate through-pitch variation especially at non-ideal defocus and exposure conditions. For the 65nm technology that we study, the through-pitch variation, after RET application, is 5nm at zero defocus (ideal focus condition) and 12nm at the maximum defocus value of 100nm. Such variations in linewidth translate to 100% and 527% variations in NMOS device leakage respectively. Fortunately, most devices have pitches that are less sensitive to defocus and the expected defocus value is smaller than the maximum.

In this work, we propose a novel detailed placement technique that changes the placement of cells to change the pitches of devices and consequently reduce their leakage. Cells can be composed of several devices; pitches of the devices that are closest to the cell boundaries (henceforth referred to as *boundary devices*) change with placement. Placement has negligible impact on the pitches of devices other than the boundary devices; this is due to their large distance from the boundary and shielding from boundary devices in the cell. However, most commonly used cells such as small- to moderately-sized buffers, inverters, NANDs, and NORs have all of their devices near boundaries. For such cells, device pitches and consequently leakage is affected by detailed placement. For example, the leakage of a NAND gate (NAND2X1) changes by 18% when it is sandwiched between two other NAND2X1 gates versus when it has no neighbors.

Our methodology involves two steps. First, a matrix is constructed to capture the leakage when two cells are placed next to each other. This matrix is used to drive our optimization and to evaluate a given placement for leakage. Second, we divide the design into small windows and optimize the windows individually. During the optimization cells are redistributed in rows, and within each row their ordering, spacing, and orientation is optimized using a traveling salesman formulation. We ensure that the timing-critical cells remain unaffected during optimization to minimize the impact on their delays and the delays of their interconnects.

A recent work by Hu et al. [13] proposed a pattern-sensitive placement approach for minimizing linewidth variation. The work by [13] was published in the period between the submission and acceptance of our work. The objective of [13] is to minimize total edge placement error (EPE)[1] by modifying detailed placement subject to wirelength constraints. Although the approach of our work and that presented by Hu et al. appear similar, they differ in several aspects. Hu

[1]EPE is a measure of linewidth variation.

ISLPED'07, August 27–29, 2007, Portland, Oregon, USA.

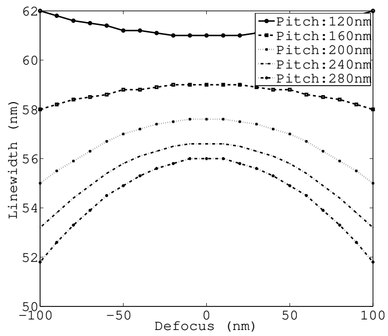


Figure 1: Variation of simulated on-silicon linewidth with defocus for different pitches. Linewidth increases with defocus for dense (small pitch) patterns, and decreases for sparse (large pitch) patterns.

et al. seek to minimize the overall EPE variation while we seek to to minimize leakage power. Hu et al. do not consider placement of filler cells[2] while we explicitly optimize their placement. Consideration of filler cells in detailed placement is very important. Filler cells in sub-90*nm* technology have non-functional poly and their placement can alter poly pitches and, consequently, device leakage as explained in Section 2.2.

The remainder of this paper is organized as follows. Section 2 gives background on systematic ACLV and detailed placement. In Section 3 we present our approach to capture context-dependent leakage in the form of a matrix. Section 4 describes our leakage optimization methodology. Section 5 presents experimental validation and Section 6 concludes.

2. BACKGROUND

ACLV is a significant contributor to leakage variation. In this section we briefly describe the role of device pitches in ACLV and the use of detailed placement to alter pitches.

2.1 Systematic ACLV

Despite the use of RETs, linewidth variation of 10-20% is typical in modern manufacturing technologies and arises largely due to lithography variations. Lithography is tuned for a set of process conditions and pitches; existence of non-ideal process conditions and different pitches causes lithography variation. A large fraction of this variation is systematic with pitch and defocus as shown in Figure 1. The figure plots interpolated foundry data *after application of RETs including OPC and scattering bar insertion* over a realistic defocus range. The quadratic dependence of linewidth on defocus has been reported and used in several previous works [16, 7, 9]. We have found linewidth for multiple devices with similar pitch and defocus to be nearly identical in foundry data. Therefore, we use the plot in Figure 1 to predict linewidth given pitch and defocus.

From the plot we note that dense lines have a larger linewidth than sparse lines across all defocus values. Also, sparser lines exhibit a larger linewidth decrease with defocus. This systematic nature of ACLV has been exploited in recent works for timing analysis [9], design robustness [10], and leakage analysis and systematic-variation aware linewidth biasing [14]. Since leakage increases exponentially as linewidth decreases, devices with dense lines will have significantly less leakage than other devices across all defocus values. This observation is the basis for the proposed optimization. We also observe that the linewidth change due to pitch saturates (as can also be seen in Figure 1) as the pitch approaches the *optical radius*. Optical radius is the radius of influence due to proximity; features near or beyond the optical radius due not affect linewidth.

Kahng et al. [14] presented a systematic-ACLV aware leakage analysis and optimization methodology. Their approach calculates the pitches of all cells in the design and finds the susceptibility of the cells to defocus-induced linewidth variations. The cells that are more susceptible are preferred for optimization over the others in a previously proposed leakage optimization [12]. Our approach modifies the pitches themselves to reduce leakage and is complementary to theirs. Gupta et al. [11] proposed a placement perturbation approach to increase the number of scattering bars that can be inserted. Scattering bars are very thin, non-printing lines that assist printability by reducing through-pitch variation. While [11] can increase the number of scattering bars and reduce through-pitch variations, design objectives such as delay and leakage are not targeted. The approach of [11] is also limited to perturbing cells in the neighboring free space in a single row, which limits the opportunities for optimization. The use of detailed placement to enhance the printability of cells has also attracted interest from the industry recently [8].

[2]Filler cells are placed in the empty space between actual cells to maintain power and ground rail connectivity.

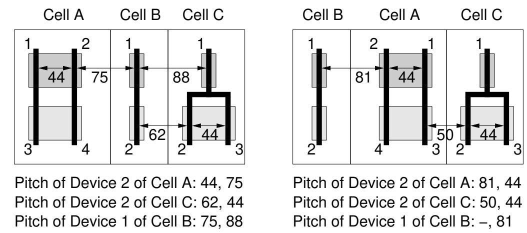


Figure 2: Detailed placement affects device pitches. Two placements of three cells in a row and the device pitches are shown.

2.2 Detailed Placement

Traditionally, placement is separated into two phases – global placement and detailed placement. Global placement generates a legalized (i.e., with no overlaps) placement of standard cells in rows. Detailed placement is a refinement step which performs small-range perturbations to generate a new optimized placement. Several approaches to detailed placement have been proposed with most focusing on wirelength minimization (e.g., [6]) or timing [5]. Our approach, to the best of our knowledge, is the first to consider the impact of detailed placement on poly gate pitch to reduce leakage which is strongly and systematically dependent on pitch.

Placement can change the pitches of boundary devices of a cell by using the following three knobs:

- Neighbor selection. Different cells have different spacings between the boundary and the boundary devices. So the neighbor of a cell affects the pitch of its boundary devices.
- Orientation. Within a cell the spacing between the left boundary and the closest boundary devices is different from that between the right boundary and the boundary devices closest to the right boundary. So, the orientation of a cell (i.e., "flipped" or not) affects the pitches of the boundary devices.
- Cell-to-cell spacing. In general introducing space between two cells causes the pitches of the boundary devices of the cells to become sparse. However, in current technologies, fillers (which are always inserted in the space between any two cells) can have non-functional polys that can increase the pitches of the boundary devices in the neighboring cells. So, cell-to-cell spacing affects the pitches of the boundary devices irrespective of the fillers containing polys.

Figure 2 shows two placements of five cells in a row and how the pitches of the gates in the cells change.

Fillers are inserted between separated cells in a placement to ensure connectivity of the power and ground rails in the space between the two cells. In 65*nm* and beyond technologies, fillers may have non-functional polys to enhance layout uniformity. Such fillers decrease the pitch of the devices in neighboring cells (i.e., make the pitch dense). On the other hand, fillers that do not have polys increase the pitches of the devices in neighboring cells (i.e., make the pitch sparse). In both cases filler insertion is a powerful knob to control device pitches and is considered in our approach.

3. ASSESSING LEAKAGE IMPACT

Potential leakage savings from detailed placement depend on the following factors.

- Pitch range. The minimum pitch attainable by detailed placement depends on the boundary device to boundary spacing of the cells. When fillers contain polys, the maximum pitch is attained when two cells with large boundary device to boundary spacing are placed next to each other. If fillers do not have polys, the maximum pitch is attained when fillers are inserted between cells. Larger difference between minimum and maximum attainable pitches affords greater leakage reduction. For our 65nm library, the spacing between neighboring gates of any two cells when they abut varies between 210nm and 520nm.
- Linewidth variation due to pitch. This is process dependent. We expect larger leakage reduction if the linewidth variation due to pitch is large. For our 65nm process, linewidth is 60nm for {210nm, 210nm} pitch (where first and second distances in the tuple are left and right spacings with the immediate neighbors respectively) and 56nm for {520nm, 520nm} pitch at 0nm defocus. At 100nm defocus, the linewidth is 60nm and 51nm for the two pitches, respectively.
- Leakage reduction with linewidth. Leakage decreases exponentially with linewidth increase. Larger leakage change with linewidth change allows more leakage optimization by detailed placement. For our technology, PMOS and NMOS leakages increase from 0.383$\mu A/\mu m$ and 0.270$\mu A/\mu m$ to 1.868$\mu A/\mu m$ and 0.887$\mu A/\mu m$ respectively when linewidth decreases from 60nm to 51nm.

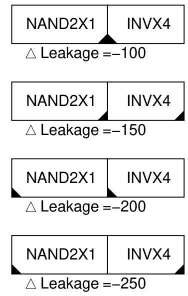
We construct a Δ*leakage matrix L* to capture the leakage change when a cell is placed next to another cell, with respect to when it is placed without any neighbor. L additionally needs to capture the fact that the leakage change depends on which of the two sides of the two cells touch. Thus the matrix has two rows corresponding to the two sides of each cell. The matrix needs to be constructed only once for a library; if there are N cells in the library, the matrix has $2N$ rows and columns. We use *Side 0* and *Side 1* to denote the left and right sides of a cell, respectively.

$$L_{ij} = \Delta leakage_{\lfloor i/2 \rfloor} + \Delta leakage_{\lfloor j/2 \rfloor}$$

when Side $i\%2$ of Cell $\lfloor i/2 \rfloor$ touches Side $j\%2$ of Cell $\lfloor j/2 \rfloor$. $\Delta leakage_{\lfloor i/2 \rfloor}$ is the leakage change of Cell $\lfloor i/2 \rfloor$ with respect to when it has no neighbors.

Calculation of leakage when two cells abut consists of the following two parts:

1. Linewidth calculation. We use the Bossung plot in Figure 1, which captures the systematic through-pitch linewidth variations, to calculate linewidth from pitch and defocus. Device pitches can computed from device to boundary spacings for all devices in the two cells as described in [14]. Defocus depends on the process conditions and is difficult to predict; so we assume it to be a random variable with nominal distribution ($\mu = 0nm, \sigma = 33.3nm$). We use the calculated pitch value and the defocus distribution to find the distribution of linewidth. Lithography simulation can alternatively be used to generate a more accurate linewidth distribution albeit at a higher runtime.
2. Cell leakage calculation. To calculate the device leakage distribution from the linewidth distribution, we use a leakage lookup table which is characterized with SPICE for a variety of gate width and gate length (linewidth). We then calculate the expected value of device leakages for all devices in the two cells and use them to calculate cell leakage. Our cell leakage calculation method is similar to Rao et al. [18]. Using logic propagation in the cell, we find the fraction of states in which each device leaks and call it the off-fraction. Leakage of a cell is the sum of leakages of its devices weighted by their respective off-fractions. As described in [18] this methodology is easily extensible to more accurate state-dependent leakage calculation. Given a set of linewidths for all devices in a cell, our leakage calculation methodology has an error under 2% (as also reported by Rao et al.) with respect to SPICE which may also be used for greater accuracy.



	NAND2X1_LEFT	NAND2X1_RIGHT	INVX4_LEFT	INVX4_RIGHT
NAND2X1_LEFT	−260	−240	**−100**	**−150**
NAND2X1_RIGHT	−240	−220	**−200**	**−250**
INVX4_LEFT	**−100**	**−200**	−40	−60
INVX4_RIGHT	**−150**	**−250**	−60	−80

Figure 3: Creation of Δleakage matrix L. The bold entries are filled by placing NAND2X1 and INVX4 next to each other. Other entries can be similarly found by placing NAND2X1 next to another NAND2X1 and by placing INVX4 placing next to another INVX4.

Our matrix construction methodology is fast and feasible for use for large libraries. We note that such a matrix needs to be created for the corner for which leakage analysis and optimization is desired. In our studies, we use the typical-leakage corner which is typical process, 1.1v, and 85^{o}C (PVT).

The matrix abstracts the pitch impact on leakage that arises due to through-pitch ACLV for use in optimization. Such a matrix may be created by the process engineers and library designers, and can be used by circuit designers to evaluate and optimize leakage.

4. LEAKAGE OPTIMIZATION

Given the impact of placement on leakage, we now present a detailed placement technique that minimizes leakage. A few factors make certain cells more favorable for optimization. For example, low-V_{th} cells are more favorable than standard-V_{th} and high-V_{th} because they have larger (absolute) leakage reductions; similarly, standard-V_{th} are more favorable than high-V_{th}. Cells that have smaller number of devices such as small- to moderately-sized inverters, buffers, NAND's, and NOR's are the most affected by proximity and are more favorable. Our optimizer maximizes leakage savings for such cells.

We dissect the design into small windows and run the optimization for each window. Such a method is effective in our case because of the localized nature of proximity which does not hold, for example, with total wirelength objectives. The optimization relies on getting a rich enough set of boundary-to-device spacings and whitespace to reduce leakage. Even a small window containing 15 cells offers enough scope for optimization. Smaller windows restrict the movement of cells within their boundaries and hence the wirelength increase is bounded. Moreover, smaller windows are faster to optimize and different windows may be simultaneously and independently optimized on multiple CPUs. Prior to the optimization, all fillers are removed and are inserted back after optimization in the whitespace.

To simplify the explanation of our optimization, we first assume that there is only one row in the window and that there is no space for fillers in the row. Under these assumptions, the problem is to identify an ordering of the cells, along with the cells that must be flipped to yield the maximum leakage reduction (i.e., minimize the sum of $\Delta leakage$ for all cells in the window). We transform this problem to the well-known traveling salesman problem (TSP) [15] as follows.

- We create two vertices for every cell – one for the left side and another for the right side.
- Edge weights or distances between vertices denote the leakage reduction when the sides represented by the vertices touch. These weights are obtained from the Δleakage matrix L.
- Weights of edges between vertices that denote sides of the same cell are set to $-\infty$ since the two vertices must always occur consecutively in the TSP tour.

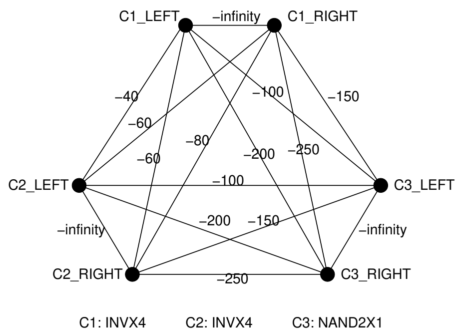


Figure 4: Creation of the graph for three cells C1, C2, and C3. C1 and C2 are of types INVX4 and C3 is of type NAND2X1. Edge weights are derived from matrix *L* of Figure 3. For edges between two vertices that belong to the same cell, a weight of $-\infty$ is assigned.

We can now use a standard symmetric TSP heuristic capable of handling large negative weights to get a TSP tour. The order of vertices in the tour gives the order in which cells must be arranged in the row and their orientations for maximum leakage reduction. We solve the TSP with the *multifragment greedy* algorithm [3] that considers edges in order of their weight to be inserted into the tour.

Space Allocation for Fillers. We now lift the assumption that no space is available for filler cells. So the optimization must additionally space the cells and insert fillers to minimize leakage. Towards this, we calculate the leakage reduction of all cells in the library when a filler cell of the minimum width (FILLX1) touches an edge of the cell. The matrix *L* is expanded by one row and one column for the FILLX1. We assume that proximity effects do not surpass the filler of minimum width and therefore the leakage of cell does not depend on the size of the filler. This assumption is valid when fillers contain dummy polys as well as when they do not. When fillers contain polys, the separation between a boundary poly and the boundary is identical for all fillers. This is generally true because fillers other than the smallest filler are essentially abutting copies of the smallest filler. So the pitch of a neighboring cell is same irrespective of the filler size. When fillers do not contain polys, the width of the smallest filler is typically sufficient to keep the devices of the next cell outside the optical radius. Larger fillers push the device further but devices beyond the optical radius have negligible proximity impact so all fillers have nearly identical pitch impact.
Our graph for TSP requires the following changes:

- We add vertices corresponding to FILLX1's; the number of added vertices is equal to the number of FILLX1's insertable in the row.
- The weight of the edges between fillers is set to zero and the weight of the edges between fillers and cells are obtained from the Δleakage matrix *L*.

For example, if space is available for 20 FILLX1's, we add 20 new vertices to the graph, set the weight of the edges between them to zero, and set the weights of the edges between a vertex representing an side of a cell and all vertices representing fillers from the matrix *L* to reflect the leakage reduction when the cell's side touches a filler.

As before, we solve a symmetric TSP with the multifragment greedy heuristic. If two fillers are consecutive in the tour, two FILLX1's must be placed next to each other. Two abutting FILLX1's are identical to a FILLX2; so we replace multiple consecutive fillers with larger fillers. We evaluate the quality of our TSP-based single-row placement solution against an optimal solution found by enumerating all possible single-row placement solutions for two arbitrary cells. Table 1 compares the leakage results normalized against the maximum leakage (which is also found by enumeration). Our approach is consistently able to attain near-optimal solutions with significantly less runtime for other configurations as well.

Multiple Rows. We now eliminate the assumption that there is only one row in the window. We exhaustively partition the set of cells into the rows and optimize these partitions using the single-row optimization. The number of partitions can be computed as a sum of Stirling numbers of the second kind and is nearly exponential with the number of cells in the window. However, in reality a large number of these partitions can be pruned due to row capacity constraints and because of multiple instances of the same cell master (which are alike) in the window. Further, best single-row results for some rows can be cached during the partitioning. With these runtime improvements, our approach handles up to two rows with ease, and three rows with feasible runtime assuming 20 cells in the window.

Table 1: Leakage comparison of TSP-based placement against optimal found by enumerating all placements. Leakage normalized against maximum leakage. Cell set 1 is {INVX1, INVX1, INVX1, NAND2X1, NAND2X1, AOI22X1, AOI22X1} and cell set 2 is {INVX2, INVX2, NOR2X0, NOR2X0, NOR2X0, MUX2X0, MUX2X0, MUX2X0}

Cell Set,	Max.	Optimal		TSP-based	
#Fillers	Leakage	Leakage	CPU (s)	Leakage	CPU (s)
1, 0	1	0.928	0.22	0.932	0.030
1, 5	1	0.804	270.18	0.806	0.034
2, 0	1	0.976	1.08	0.976	0.033
2, 3	1	0.922	221.09	0.922	0.034

Minimizing Timing Impact. The perturbation of detailed placement from the original placement results in wirelength change, which can impact wire parasitics and consequently timing. Even though our localized placement perturbations do not significantly affect timing, small changes in the timing of critical paths can affect the minimum clock cycle time. To minimize the timing change of critical paths, we fix the cells and nets in the critical paths: fixed cells are not moved during optimization and fixed nets are not changed during engineering change order (ECO) routing that is performed after optimization. Since the nets in the critical path are fixed, all cells connected to these nets should also be marked as fixed and not moved during optimization. We note that the delay of such nets can marginally change due to the coupling capacitance with neighboring nets, the routing for which may change. We also fix all flip-flops, clock buffers, and clock nets to avoid any impact on the clock tree.

During optimization, for each cell marked as fixed, we break the row in which the cell is placed into two parts: left of the cell and right of the cell. The two parts are optimized individually; this ensures that the fixed cells do not move and no other cells overlap with the fixed cells. Although we do not move fixed cells during optimization, our approach considers their location during the placement of other cells. The overall algorithm of our optimization is presented in Figure 5. Given an original placement, list of critical cells, Δleakage matrix *L*, and a window size, the optimization outputs a final placement with lower leakage.

Minimizing Wirelength Increase. Wirelength increase is undesirable because it can cause congestion and degrade routability. Also, wires act as capacitive elements and longer wires can increase dynamic power. We expect a smaller window size to cause smaller wirelength increase because the movement of cells during optimization is restricted to within the windows. To reduce wirelength we run the optimization in phases with each phase successively increasing the window size until a *final window size*, which is a user input, is attained. The result of a phase is only accepted if it improves upon the result of the previous phase by more than a threshold (set to zero in our experiments). This policy has the effect that cells are moved farther only if the leakage reduction is greater than the threshold.

5. EXPERIMENTAL STUDY

In this section, we discuss the details of our experimental setup for optimization followed by detailed routing, and present results.

5.1 Experimental Setup

A high-level overview of our experimental setup is show in Figure 6. For setting up optimization flow, we first perform synthesis

Input: Placed design, timing critical cells, δleakage matrix L (of Figure 1) that denotes leakage change when any two cells touch, window size
Output: New placement with lower leakage and small/no delay impact

1 $D \leftarrow$ {critical cells} $\cup$ {cells connected to output nets of critical cells}
2 **forall** windows w in the design
2.1 $C \leftarrow$ {All cells in w}
2.2 PartitionAndPlace($C - D$, w)

PartitionAndPlace(C, w)
1 r = firstRowOf(w)
2 $bestCost = \infty$
3 **forall** $S \in$ Subsets(C)
3.1 **if** (rowCapacity(r) $< \sum_{c \in S}$width(c)) // row capacity not exceeded
3.1.1 $<tour, Cost_r>$ = TSPPlace(S, r) // Place cells S in row r
3.1.2 $Cost_{w-r}$ = PartitionAndPlace($C - S$, $w - r$) // Place remaining cells in remaining rows
3.1.3 **if** (bestCost $> Cost_r + Cost_{w-r}$)
3.1.3.1 $bestCost = Cost_r + Cost_{w-r}$
3.1.3.2 $bestTour = tour$
4 save(r, $bestTour$)
5 return $bestCost$

TSPPlace(C, r)
1 F = width(r) - $\sum_{c \in C}$width(c) // Number of fillers
2 $G = (V, E)$; $V = C \cup_1^F$ {"FILL"}; Construct E from L // Insert cells in C and F fillers into V
3 Solve TSP on G
4 return <tour from TSP, cost of tour>

Figure 5: **Detailed placement flow for leakage optimization.**

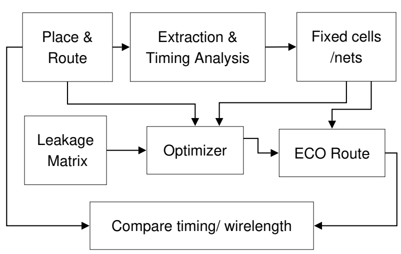


Figure 6: **Our experimental flow.**

of testcases using multiple threshold voltage libraries (high-V_{th} and normal-V_{th}) in 65nm technology. We then perform placement, detailed routing followed by extraction and timing analysis. From the timing analysis result, we identify all timing critical cells for feeding the optimizer. We also create a set of nets corresponding to the critical cells that should not be touched during ECO routing of the optimized detailed placement.

The optimizer reads in the placed and routed design, *fixed cells* list, leakage matrix L and performs placement optimization. The optimizer outputs a legal layout (in which the cell orientations are correct and no cell overlaps with any other cell). We then perform ECO routing on the optimized result along with fixed net objective and then perform parasitic extraction and timing analysis.

The most important steps in our experimental setup are leakage matrix construction and interfacing the optimizer to the router. These steps are discussed in greater detail in the following.

Leakage Matrix Construction. Bossung LUT provides linewidth of devices in different layout contexts. To construct the Bossung LUT, we take as input a standard cell layout in 65*nm* technology and compute poly pitch for all the devices in the layout. This is performed by analysis of neighborhood of each device in the layout. We use a device layout analysis tool (built on the OpenAccess API [1]) to search the neighborhood of every device and compute the spacing to other devices. To obtain the printed linewidths for devices at different pitches, we take as input their litho-simulated device contours (generated *after application of scattering bars and OPC*) at specific defocus values. We interpolate between defocus values using a 2^{nd}-degree polynomial [16]. We use the procedure outlined in the steps above to obtain linewidths of devices in different cell layout contexts for construction of the leakage matrix. We assume a normal distribution for defocus with a mean of 0*nm* and σ of 33.3*nm*.

Table 2: **Testcases used in experimental validation.**

Circuit	#cells	Max. Speed (MHz)	Leakage (mW)	Dynamic (mW)	Wirelength (mm)
AES (80% util.)	18665	413	0.6211	0.4002	330.87
AES (85% util.)	18726	419	0.6323	0.4127	320.65
DES	79419	417	6.3083	3.6754	1146.14

Optimizer – ECO Route Interface. To optimize a given detailed placement for minimum leakage, we start from an existing placed and detailed routed design. We perform timing analysis on the design to identify all timing critical paths. We then choose all cells from critical paths that have a slack value ranging from the minimum value to 5% of the clock cycle time and mark them as fixed. Since the nets connected to these fixed cells also cannot move, we create a *dont_touch* list of nets connected to all fixed cells. All route segments corresponding to fixed nets are not moved during ECO routing. To prevent disruptions to the routing of these nets, we update the existing fixed cells list to include all cells connected to fixed nets.

We use *Cadence RTL Compiler (v5.1)* for multi-V_{th} synthesis of our testcases. For our experiments, we use the 50 most-frequently used cells from high-V_{th} and nominal-V_{th} libraries. To run placement, clock tree synthesis, routing and timing analysis, we use *Cadence SOC Encounter (v5.2)*. The details of our testcases are shown in Figure 2. The standard cell row utilization for our testcases: aes and des (available from *opencores.org* [2]) are 80% and 73% respectively. To demonstrate leakage reduction at higher utilizations, we implemented aes at 85% row utilization. Row utilizations $>$ 80% are not common because of routing congestion concerns. We built our optimizer on top of *OpenAccess API (v2.2.4)*.

The inputs to our optimizer are routed design and a list of fixed cells. The output from the optimizer is the design with modified placement with "dangling" wires for some cells (since locations of cells are perturbed during optimization). We feed the output from the optimizer to SOC Encounter and invoke the router in ECO mode along with *dont_touch net* router directives[3]. After ECO routing, we perform parasitic extraction and timing analysis to evaluate change in wirelength and timing. We use the worst-case corner for timing analysis in our flow.

5.2 Results

We evaluate the proposed approach for leakage reduction and change in wirelength, delay and dynamic power. Table 3 presents our results for the two testcases and multiple window sizes. The leakage reduction upperbound indicates the maximum leakage savings possible if the lowest-leakage context for all cells could be created by choosing their neighbors. The upperbound may not be attainable because only the cells available in the window can be used as neighbors and because of limited availability of free space. The leakage reduction, wirelength change, and delay change are with respect to the original placed and routed design. Wirelength is the actual routed wire length after detailed routing; in all cases detailed routing finished without any violations.

From the results, window sizes of $6\mu\times$ 2 rows and $4\mu\times$3 rows offer good solution quality with feasible runtime. We observe that the affect of the optimization on maximum frequency is marginal. For the testcases AES and DES, 10.97% and 23.57% cells were marked to be fixed during optimization. Without marking these cells as fixed, the maximum frequency dropped by 5.62% for AES while the leakage reduction only increased from 6.41% to 7.45%. While the affect of the optimization on wirelength is not negligible, without our wirelength reduction policies the wirelength increase was 12.33% for AES in comparison to 8.14% with the policies. The leakage reduction is smaller at higher utilization because of lower availability of whitespace (which is favorable to leakage reduction without experi-

[3]To direct SOC Encounter to honor existing routing of fixed nets, we use "setAttribute -net $< NET\_NAME >$ -skip_routing true" command prior to invoking ECO route

Table 3: **Assessment of impact on leakage, wirelength, and delay with the proposed technique. The row annotated with † symbol corresponds to results without the use of delay and wirelength reduction policies.**

Circuit	Leakage Reduction Upperbound	Final Window Size	Proposed Technique: Leakage Reduction (%)	Wirelength Impact (%)	Max. Frequency Impact (%)	Dynamic Power Impact (%)	Runtime (s)
AES	8.95%	$4\mu\times1$ row	2.91	+0.72	+0.33	+0.13	5.18
		$6\mu\times1$ row	4.16	+2.39	-0.41	+0.31	8.72
80%		$8\mu\times1$ row	5.08	+4.94	-1.18	+0.45	14.64
utilization		$4\mu\times2$ rows	5.21	+3.86	+0.50	+0.36	37.90
		$6\mu\times2$ rows	6.41	+8.14	-0.49	+0.61	301.35
		$2\mu\times3$ rows	4.02	+2.08	+0.46	+0.25	23.83
		$4\mu\times3$ rows	6.44	+7.12	-0.41	+0.67	1964.09
		$6\mu\times2$ rows†	7.45†	+12.33†	-5.62†	+0.92†	284.34†
AES	9.50%	$4\mu\times1$ row	1.81	+0.93	+0.21	+0.12	5.23
		$6\mu\times1$ row	2.77	+2.65	-0.33	+0.22	9.57
85%		$8\mu\times1$ row	3.57	+5.08	-0.91	0.41	18.01
utilization		$4\mu\times2$ rows	3.64	+4.03	+0.63	+0.33	50.99
		$6\mu\times2$ rows	4.82	+8.15	-0.52	0.60	533.19
		$2\mu\times3$ rows	2.56	+2.51	-0.11	0.21	24.13
		$4\mu\times3$ rows	4.76	+7.22	-0.56	0.51	2983.56
DES	8.30%	$4\mu\times1$ row	4.85	+3.53	-0.62	0.05	15.00
		$6\mu\times1$ row	6.04	+5.83	-0.87	0.06	22.25
73%		$8\mu\times1$ row	6.48	+7.49	-0.58	0.07	28.64
utilization		$4\mu\times2$ rows	6.28	+6.06	-0.37	0.06	51.32
		$6\mu\times2$ rows	6.76	+8.42	-0.50	0.07	180.98
		$2\mu\times3$ rows	5.70	+5.37	-0.54	0.05	51.71
		$4\mu\times3$ rows	6.79	+7.76	-0.62	0.07	1764.35

mental setup) and because of fewer opportunities to move cells across rows. The wirelength increase did not affect dynamic power significantly because wire capacitance is a small fraction of total capacitance. Furthermore, wires do not contribute to short-circuit power. We expect the dynamic power impact to be even smaller when extensive clock gating is used.

We consider the leakage reduction results encouraging. Unfortunately, the leakage reduction upperbound is small in our experiments. This is because the placement tool used by us inserts fillers next to most cells in the design and fillers, which have dummy polys that increase pitch, are leakage favorable in our process. We expect much larger leakage reduction if: (1) fillers do not contain dummy polys such as in 90*nm* technologies, (2) process has an opposite linewidth change as ours, i.e., dense lines print narrower, or (3) placer does not abut fillers to most cells in the design.

6. CONCLUSIONS

Through-pitch ACLV causes significant variation in leakage and consequently the leakage of cell is substantially affected by the neighbors of the cell. Fortunately, these variations have a large systematic component and the leakage of a cell can be predicted more accurately once its neighbors are known. We presented a methodology to construct a matrix that captures the leakage of a cell under various placement contexts.

We have proposed a detailed placement approach that arranges cells in standard cell rows and redistributes whitespaces such that the leakage of the cells is minimized. In doing so, the optimization attempts to minimize the leakage for the cells that offer the most leakage savings such as lower V_{th} cells and smaller cells since their leakage is most affected by context. We fix the timing critical cells and their interconnects to minimize timing impact and run the optimization over progressively increasing window sizes to minimize wirelength increase. The optimization considers all feasible ways to distribute the cells in the available rows and a TSP-based optimizer places the cells in each row. We have assessed the TSP-based optimizer to be of near-optimal quality and because all feasible ways to distribute the cells in rows are considered, our final placement in the window is near-optimal.

We evaluated our technique for three testcases with industry tools and process information. Our results indicate leakage reduction to be in the range of 5%-7% for 7%-8% wirelength increase, and negligible delay and dynamic power impact. We hypothesize that in technologies in which fillers do not contain dummy polys or in which the process response to pitch variations is opposite to ours, higher leakage reductions would be attained. Our ongoing work is in two directions: (1) evaluation of the technique for other technologies, and (2) use of detailed placement to improve timing robustness.

7. REFERENCES

[1] OpenAccess API. http://openeda.si2.org/

[2] http://www.opencores.org

[3] J. L. Bentley. Experiments on Traveling Salesman Heuristics. In Proc. Symp. Discrete Algorithms, pages 91–99, 1990.

[4] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi and V. De. Parameter Variations and Impact on Circuits and Microarchitecture. In *Proc. of the ACM/IEEE Design Automation Conf.*, pages 338–342, 2003.

[5] A. Chowdhary, K. Rajagopal, S. Venkatesan, T. Cao, V. Tiourin, Y. Parasuram and B. Halpin. How Accurately Can We Model Timing in a Placement Engine? In *Proc. of the ACM/IEEE Design Automation Conf.*, pages 801–806, 2005.

[6] K. Doll, F. M. Johannes and K. J. Antreich. Iterative Placement Improvement by Network Flow Methods. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 13(10):1189–1200, 1994.

[7] D. G. Flagello, H. van der Laan, J. van Schoot, I. Bouchoms and B. Geh. Understanding Systematic and Random CD Variations Using Predictive Modelling Techniques. In *Proc. of the SPIE Conf. on Optical Microlithograhpy*, pages 162–175, 1999.

[8] P. Groeneveld. Manufacturability Driven Physical Synthesis. In *IEEE Intl. Workshop on Design for Manufacturability and Yield*, invited talk, 2006.

[9] P. Gupta and F.-L. Heng. Toward a Systematic-Variation Aware Timing Methodology. In *Proc. of the ACM/IEEE Design Automation Conf.*, pages 321–326, 2004.

[10] P. Gupta, A. B. Kahng, Y. Kim and D. Sylvester. Self-Compensating Design for Focus Variation. In *Proc. of the ACM/IEEE Design Automation Conf.*, pages 365–370, 2005.

[11] P. Gupta, A. B. Kahng and C.-H. Park. Detailed Placement for Improved Depth of Focus and CD Control. In *Proc. of the Asia and South Pacific Design Automation Conf.*, pages 343–348, 2005.

[12] P. Gupta, A. B. Kahng, P. Sharma and D. Sylvester. Selective Gate-Length Biasing for Cost-Effective Runtime Leakage Control. In *Proc. of the Design Automation Conf.*, pages 327–330, 2004.

[13] S. Hu and J. Hu. Pattern Sensitive Placement for Manufacturability. In *Proc. of the Intl. Symp. on Physical Design*, pages 27–34, 2007.

[14] A. B. Kahng, S. Muddu and P. Sharma. Defocus-Aware Leakage Estimation and Control. In *Proc. of the Intl. Symp. on Low Power Electronics and Design*, pages 263–268, 2005.

[15] E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan and D. B. Shmoys. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Wiley Interscience Series in Pure and Applied Mathematics, 1985.

[16] C. A. Mack and J. D. Byers. Improved Model for Focus-Exposure Data Analysis. In *Metrology, Inspection and Process Control XVII, SPIE Vol. 5038*, pages 396–405, 2003.

[17] S. Postnikove and S. Hector. ITRS CD Error Budgets: Proposed Simulation Study Methodology. May 2003.

[18] R. M. Rao, J. L. Burns, A. Devgan and R. B. Brown. Efficient Techniques for Gate Leakage Estimation. In *Proc. of the Intl. Symp. on Low Power Electronics and Design*, pages 100–103, 2003.