Diffusion Break-Aware Leakage Power Optimization and Detailed Placement in Sub-10nm VLSI

Sun ik Heo[†], Andrew B. Kahng[‡+], Minsoo Kim[‡] and Lutong Wang[‡]

[+]CSE and [‡]ECE Departments, UC San Diego, La Jolla, CA, USA
[†]Samsung Electronics Co., Ltd., Hwaseong-si, Gyeonggi-do, South Korea
{abk, mik226, luw002}@ucsd.edu, sunik.heo@samsung.com

ABSTRACT

A *diffusion break* (DB) isolates two neighboring devices in a standard cell-based design and has a stress effect on delay and leakage power. In foundry sub-10nm design enablements, device performance is changed according to the type of DB – *single diffusion break* (SDB) or *double diffusion break* (DDB) – that is used in the library cell layout. Crucially, *local layout effect* (LLE) can substantially affect device performance and leakage. Our present work focuses on the 2^{nd} *DB effect*, a type of LLE in which distance to the second-closest DB (i.e., a distance that depends on the placement of a given cell's *neighboring* cell) also impacts performance of a given device. In this work, we implement a 2^{nd} DB-aware timing and leakage analysis flow, and show how a lack of 2^{nd} DB awareness can misguide current optimization in place-and-route stages. We then develop 2^{nd} DB-aware leakage optimization and detailed placement heuristics. Experimental results in a scaled foundry 14nm technology indicate that our 2^{nd} DB-aware analysis and optimization flow achieves, on average, 80% recovery of the leakage increment that is induced by the 2^{nd} DB effect, without changing design performance.

ACM Reference Format:
Sun ik Heo[†], Andrew B. Kahng[‡+], Minsoo Kim[‡] and Lutong Wang[‡]. 2019. Diffusion Break-Aware Leakage Power Optimization and Detailed Placement in Sub-10nm VLSI. In *ASPDAC '19: 24th Asia and South Pacific Design Automation Conference (ASPDAC '19), January 21–24, 2019, Tokyo, Japan.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3287624.3287679

1 INTRODUCTION

With aggressive lateral scaling of advanced CMOS technology, many challenges of standard cell architectures, floorplan, placement, routing, and timing signoff have been introduced into leading-edge IC design. Design today is challenged by numerous complex front-end-of-line (FEOL) and back-end-of-line (BEOL) design rules and layout restrictions. Moreover, aggressive area reduction via the standard-cell architecture leads to a breakdown of the traditional assumptions of composability and independence of cell-based layout. Notably, in sub-10nm VLSI, a given cell instance's timing and power can be strongly affected by the cell's neighbors in the placement; this is commonly referred to as *local layout effect* (LLE). Since standard-cell library models are provided for the cell itself, without regard to its neighbors in the placement, LLE induces performance modeling errors and/or added margins in signoff. Today's advanced-node design and signoff technologies must now carefully take LLEs into account.

New standard cell architecture with diffusion breaks. A *diffusion break* (DB) isolates two adjacent devices and is one of several critical aspects related to LLEs. For example, Slide 44 of the DAC-2018 tutorial [13], referring to such sources as [1], notes how continuous diffusion and diffusion breaks are an important lever for shrinking standard-cell area – at the cost of more complex design rules and LLEs. In modern sub-10nm enablements, there are two types of diffusion breaks: *single diffusion break* (SDB) and *double diffusion break* (DDB). SDBs occupy a smaller space than DDBs do, but incur greater LLE. On the other hand, while DDBs require more area and incur more leakage than SDBs, they make devices faster and are more immune to layout context. In this work, we ignore the 2^{nd} DB effect for DDB cells due to its negligible impact on leakage and timing. We study two types of standard cells, i.e., *SDB cells* and *DDB cells.* An SDB (resp. DDB) cell has an SDB (resp. a DDB) at both its left and right edges. In the following, we call a design with only SDB cells as *Type-I*, and a design with both SDB and DDB cells as *Type-II.*

Figure 1 shows two legal placements with SDB and DDB cells. SDB and DDB cells can be placed within a single block as long as: (i) SDB (resp. DDB) cells all align to SDB (resp. DDB) grids, where the SDB grid has a half contacted poly pitch (CPP) offset from the DDB grid; and (ii) there is a process-specific minimum spacing requirement ($Spacing_{SD}$ in Figure 1) between neighboring SDB and DDB cells. In Figure 1, for a device in a green circle, D1 is the distance to the first placed SDB, and D2 is the distance between the first placed SDB and the second placed SDB. That is, D2 refers to the "2^{nd} *DB distance* of an SDB cell", and we use the term "2^{nd} DB effect" to indicate the threshold voltage (Vt) shift due to D2 for an SDB cell. Note that the Vt shift due to the 2^{nd} DB effect causes a timing path with fixed cell instances and routing to be impacted by the *neighbors* of its cell instances. Since an SDB cell can have a second placed SDB on each of its left and right sides, in this work, we use the distance to the farther of the two second placed SDBs as the D2 of a cell.

Design impact from the 2^{nd} DB effect. Table 1 shows results of a motivating analysis that demonstrates how much the 2^{nd} DB effect

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ASPDAC '19, January 21–24, 2019, Tokyo, Japan
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6007-4/19/01…$15.00
https://doi.org/10.1145/3287624.3287679

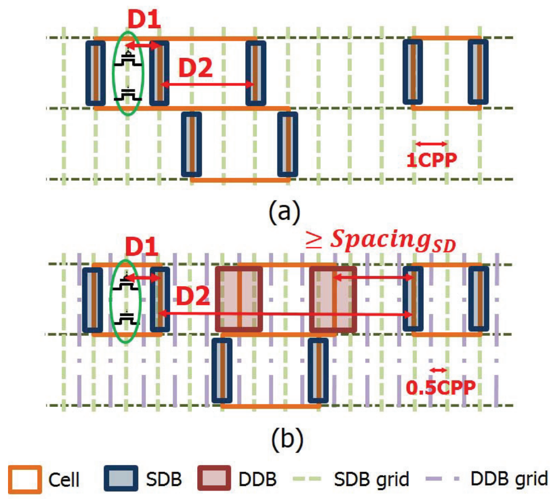

Figure 1: Examples illustrating DB types and placement constraints. (a) Layout with four SDB cells *(Type-I)*. (b) *Mixed-DB* layout with three SDB cells plus one DDB cell *(Type-II)*. The examples show the placement grid for each DB and a spacing constraint. The dashed lines show SDB and DDB grids. *Spacing$_{SD}$* **denotes the minimum spacing requirement between neighboring SDB and DDB. The delay and leakage of devices in the green circles change according to D2. D2 only considers the distance between the first placed SDB (which is intrinsic to the cell itself) and the second placed SDB (which depends on the placement of the neighbor cell). Only SDB cells experience a significant 2^{nd} DB effect.**

changes timing results and leakage power of designs.[1] We use two designs, AES and VGA, from OpenCores [25] and a reference flow of *Cadence Innovus Implementation System v17.1* [22] to generate post-P&R designs. We note that filler cell insertion is a critical step that determines 2^{nd} DB distances of cells and the impact of the 2^{nd} DB effect. Filler cells are inserted into empty spaces after place-and-route (P&R), typically starting with larger-width filler cells and finishing with smaller-width filler cells so as to greedily minimize instance count. However, filler cells with large width can create large 2^{nd} DB distances for any neighboring non-filler cells. So, in order to see the design impact of the 2^{nd} DB effect, we compare two filler cell insertion strategies: largest to smallest filler *(Standard)* and small filler only *(Small)*. As seen in Table 1, when we perform 2^{nd} DB-aware analysis for designs which come from a commercial P&R tool, 2^{nd} DB-aware leakage is larger than 2^{nd} DB-unaware leakage by 8.8% to 144.4%. In other words, actual leakage can be much larger than the leakage reported by signoff analysis that does not take into account the 2^{nd} DB effect. Further, after 2^{nd} DB-aware timing analysis, setup worst timing slacks are improved or similar, and hold worst timing slacks are not changed, due to small 2^{nd} DB impacts for hold corner. Overall, this preliminary study shows that there is substantial model-hardware miscorrelation in terms of leakage power because the 2^{nd} DB effect is not comprehended in cell library models. Since the physical location of the 2^{nd} DB matters when tracing timing impacts of optimizations, the physical layout design and signoff flows should be aware of the 2^{nd} DB effect to avoid this model-hardware miscorrelation.

[1]Details of our modeling of 2^{nd} DB effect are given in Section 3.2.

Table 1: Preliminary study of the 2^{nd} DB effect. We use 80% initial utilization and 0.35ns target clock period. Design nomenclature specifies design name (AES or VGA), filler insertion strategy (Small (*sm*) or Standard (*std*)), and design type (*Type-I* or *Type-II*). WS denotes the worst slack of a design. All values are reported by a leading-edge commercial P&R tool. Columns 2–4 have pairs of values from 2^{nd} DB-unaware and 2^{nd} DB-aware analysis, respectively. Δ indicates the percentage increase in leakage when going from (incorrect) 2^{nd} DB-unaware to (correct) 2^{nd} DB-aware analysis.

Design	**Hold WS** (ns)	**Setup WS** (ns)	**Leakage** (mW) (Δ%)
AES-*sm*-I	0.108 / 0.108	-0.056 / -0.047	0.387 / 0.742 (91.7%)
AES-*sm*-II	0.113 / 0.113	-0.055 / -0.055	0.419 / 0.456 (8.8%)
VGA-*sm*-I	0.080 / 0.113	0.016 / 0.022	1.486 / 3.186 (114.4%)
VGA-*sm*-II	0.086 / 0.086	-0.002 / -0.002	1.523 / 2.439 (60.1%)
AES-*std*-I	0.108 / 0.108	-0.056 / -0.045	0.387 / 0.909 (134.9%)
AES-*std*-II	0.113 / 0.113	-0.055 / -0.054	0.419 / 0.474 (13.1%)
VGA-*std*-I	0.080 / 0.080	0.016 / 0.024	1.486 / 3.632 (144.4%)
VGA-*std*-II	0.086 / 0.086	-0.002 / -0.001	1.523 / 2.640 (73.3%)

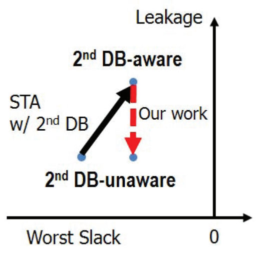

Figure 2: Target of our work. We seek to minimize *actual*, 2^{nd} DB-aware leakage power with no timing degradation.

In what follows, we focus on mitigation of analysis error – in particular, underestimation of leakage – that is caused by unawareness of the 2^{nd} DB effect. We assume a place-and-route methodology that inserts only Small fillers, to *a priori* minimize the leakage impact of the 2^{nd} DB effect. We only consider setup timing slacks during our optimization.

This work. In this work, we study design impacts caused by the 2^{nd} DB effect and propose a 2^{nd} DB-aware leakage optimization that uses cell relocation, gate sizing, Vt swapping and DB (Type) swapping to mitigate the P&R tool's lack of awareness of 2^{nd} DB effect. Our main contributions are summarized as follows.

- We study design impacts of the 2^{nd} DB effect for designs using only SDB cells (Type-I) and both SDB and DDB cells (Type-II).
- We propose a 2^{nd} DB-aware leakage optimization and placement methodology that uses relocation, gate sizing, Vt swapping and DB swapping. Our algorithm honors the placement grid for each DB and a specified spacing constraint (*Spacing$_{SD}$* in Figure 1) between SDB and DDB.

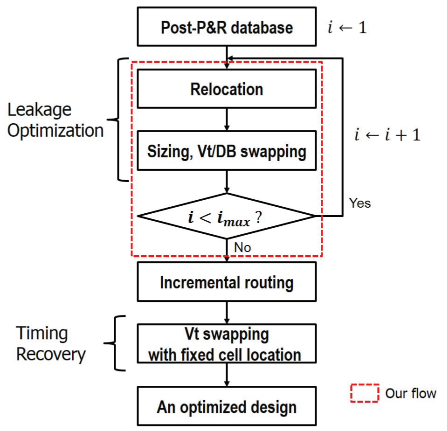

Figure 3: Overview of our overall optimization flow. The red box indicates steps that we implement. A commercial P&R tool is used for all other steps in the flow.

- We achieve 80% leakage recovery (i.e., of the leakage increment seen with correct, 2^{nd} DB-aware analysis) on average without changing design performance.

Figure 2 conceptually illustrates the target of our work. We aim to recover leakage increments caused by the 2^{nd} DB effect without any increase of negative timing slacks.

The remainder of our paper is organized as follows. Section 2 describes previous related works. Section 3 presents our problem statement, leakage model and optimization approach. Section 4 describes our experimental setup, key experiments, and results. We conclude in Section 5.

2 PREVIOUS WORK

We categorize relevant previous works as: (i) gate sizing and (ii) local layout effect and placement.

Gate sizing. Traditional gate sizing problems have been studied for many years. Objectives of the problems are to find the gate width and threshold voltage for each cell so that a circuit can achieve optimized timing, power, and area as a result. Gupta et al. [5] describe a sensitivity-based gate sizing algorithm to reduce leakage power. Hu et al. [10] and Kahng et al. [11] propose global timing recovery and leakage power reduction with a sensitivity-guided greedy sizing algorithm. Wei et al. [17] minimize total power using gate sizing and Vt assignment. Luo et al. [14] propose a combined methodology with placement and gate sizing. The work of [14] minimizes power by sequential placement, sizing, and Vt swapping stages with a slack management scheme. The works of [12][15] suggest heuristics for co-optimization of sizing and placement with minimum implant area constraints.

Local layout effect and placement. In sub-10nm, various works address LLEs from a standard cell perspective. Yang et al. [19] describe LLEs of 2^{nd} DB and gate-cut stress in 10nm high performance mobile SoCs. Zhao et al. [20] introduce gate-cut stress induced LLEs on 14nm FinFET, and Xie et al. [18] introduce SDBs and DDBs in actual FinFET devices. At the same time, many recent works aim to mitigate LLEs for detailed placement. Ha et al. [6] introduce a pre-placement methodology to mitigate LLEs from a long SDB created

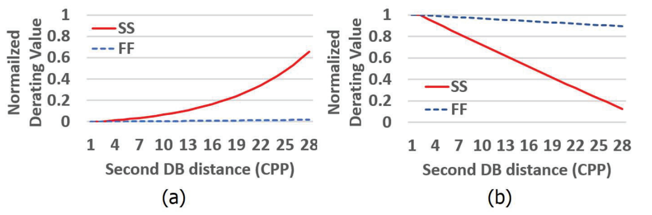

Figure 4: Normalized derating values for (a) leakage power and (b) cell delay according to the 2^{nd} DB distance.

by vertically stacked SDB cells and a 2^{nd} DB-aware timing analysis scheme. Berthelon et al. [2] propose a design and technology co-optimization with strain-induced LLEs along with asymmetric and non-rectangular active area. Han et al. [7] propose a detailed placement method to minimize the number of abutments between cells, and Du et al. [3] deal with new placement constraints of a 16nm technology. Han et al. [8] [9] also develop detailed placement algorithm to reduce the number of steps for diffusion height differences between adjacent cells. Overall, our work is distinguished from the previous works in that (i) we handle both sizing and placement with constraints from SDB and DDB, and (ii) we mitigate leakage increments caused by the 2^{nd} DB effect, which is introduced as a recent type of LLE.

3 2^{nd} DB-AWARE LEAKAGE OPTIMIZATION AND PLACEMENT

In this section, we first describe our problem statement and the timing and leakage model for the 2^{nd} DB effect. We then describe our methodology for detailed placement and leakage optimization. Figure 3 shows the overall flow of our optimization. The input of the flow is an optimized post-route design database produced by a commercial P&R tool. Our leakage optimization is performed i_{max} iterations, and is followed by incremental routing. After routing, Vt swapping with fixed cell location recovers negative timing slacks caused by routing changes. This incremental routing and Vt swapping is performed by a commercial P&R tool.

3.1 Problem Statement

Given a post-P&R design, our goal is to swap cells and perturb placement so as to minimize leakage with 2^{nd} DB awareness.

Input: A post-P&R design database from a commercial tool.

Output: An optimized design with 2^{nd} DB awareness.

Objective: To minimize leakage power of the design.

Do: Relocation, gate sizing, Vt swapping and DB swapping.

Constraints: No total negative slack (TNS) degradation, no cell overlap, grid-based placement for each DB, and a spacing constraint between SDB and DDB.

3.2 Modeling for the 2^{nd} DB Effect

Delay and leakage power of devices are changed by the 2^{nd} DB effect. We introduce our model for 2^{nd} DB using derating values of Vt shift from the work of [19] and our collaborator [21]. Figure 4 shows the normalized derating values for leakage power and delay. According to [21], we use the following sets of process, voltage, and

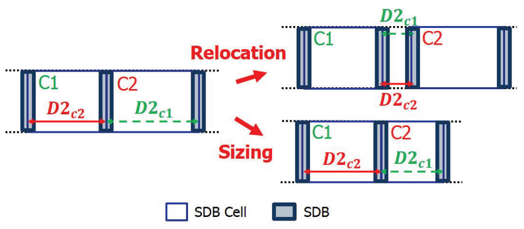

Figure 5: Examples of 2nd DB distance change due to cell relocation and sizing. The red color of $D2_{C2}$ indicates the 2nd DB distance of C2 and the green color of $D2_{C1}$ indicates the 2nd DB distance of C1.

temperature conditions: (SS, 0.72V, $-40°C$) for setup corner and (FF, 0.88V, $125°C$) for hold corner. Since the Vt shifts for PMOS and NMOS are different, we assume that the averaged Vt shift for cells is proportional to 2nd DB distance for SS and FF [21]. Advised by [21], we calculate the derating values for leakage power and delay of cells by assuming that leakage power exponentially increases with 2nd DB distance. In our library, we assume that a given DDB cell is 2CPP wider than the corresponding SDB cell, and that the 2nd DB effect maxes out at the maximum width of standard cells in our library. Furthermore, we follow design rules for the spacing constraint between neighboring SDB and DDB cells. Our derated libraries are created using a 2CPP-step of 2nd DB distance. We do not have a 2nd DB distance of 1CPP because minimum cell width is 2CPP. And, we also round down odd-valued 2nd DB distances to even values because libraries with a step size of 1CPP incur more than 3× peak memory usage during our optimization.[2]

Algorithm 1 2nd DB-aware Relocation Algorithm.

```
Procedure: $Relocation()$
Inputs: a netlist $D$, a placement of $D$
Outputs: an optimized netlist
1: $M \leftarrow \varnothing$ ; $k \leftarrow 0$
2: for all cell $c$ in the netlist do
3:     if cell $c$ is neither a DDB cell nor a flip-flop then
4:         if cell $c$ is abutted with $c.c_l$ and whitespace on right side of cell $c \geq W_{min}$ then
5:             $m_k.c \leftarrow c$ ; $m_k.m \leftarrow$ move to the right by $W_{min}$
6:             update $c.2nddb$, $c.c_l.2nddb$, $c.c_r.2nddb$
7:             $m_k.\Delta leakage \leftarrow \Delta leakage$ ; $k \leftarrow k+1$
8:         end if
9:         if cell $c$ is abutted with $c.c_r$ and whitespace on left side of cell $c \geq W_{min}$ then
10:            $m_k.c \leftarrow c$ ; $m_k.m \leftarrow$ move to the left by $W_{min}$
11:            update $c.2nddb$, $c.c_l.2nddb$, $c.c_r.2nddb$
12:            $m_k.\Delta leakage \leftarrow \Delta leakage$ ; $k \leftarrow k+1$
13:        end if
14:    end if
15: end for
16: while $M \neq \varnothing$ do
17:     Pick $m_k$ with the maximum $m_k.\Delta leakage$ in $M$
18:     Commit $m_k$ ; $M \leftarrow M \backslash m_k$
19:     if TNS degrades then
20:         Revert
21:     end if
22: end while
```

[2] When the 2nd DB-unaware leakage is 0.225mW for the AES Type-I design (60% utilization, 0.5ns clock period), the 2nd DB-aware leakage with the 1CPP step size is 0.319mW, and that with the 2CPP step size is 0.295mW; our optimization results (leakage recovery) are 99.0% and 97.1%, respectively.

3.3 2nd DB-Aware Relocation

We first describe our sensitivity-based method to *relocate* cells with no TNS degradation. Cell location is critical for the 2nd DB effect. Figure 5 illustrates how 2nd DB distance varies due to cell relocation and sizing. In this figure, $D2_{C2}$ denotes the 2nd DB distance of cell C2 and $D2_{C1}$ denotes the 2nd DB distance of cell C1. If C2 is relocated to the right by 2CPP, then the 2nd DB distances of both C1 and C2 decrease to 2CPP. If C2 is sized down and still abuts C1, then only $D2_{C1}$ decreases. Therefore, we can reduce the 2nd DB distance by cell relocation, and recover leakage power. Algorithm 1 shows this *relocation* flow (Table 2 gives notations that we use.) Line 1 initializes an empty set of candidate moves. In Lines 2–15, we calculate candidate moves per cell and the sensitivity per move. Here, we only consider moves of SDB combinational cells, as shown in Lines 2 and 3. In Lines 4–8, for each cell, we add a candidate right move (relocated to the right) and calculate the sensitivity (leakage recovery due to the move) if the cell has no whitespace on the left but whitespace on the right. Similarly, in Lines 9–14, we add a candidate left move and calculate the sensitivity. In Lines 16–21, we greedily commit the candidate move with the highest sensitivity as long as there is no TNS degradation and the placement is legal.

Table 2: Notations.

Notation	Meaning
$m_k.c$	cell for the k^{th} candidate
$m_k.m$	method for the k^{th} candidate
$m_k.sf$	sensitivity for the k^{th} candidate
$m_k.\Delta leakage$	$\Delta leakage$ for the k^{th} candidate
$\Delta leakage$	$\Delta leakage$ of a design
M	set of candidates
$c.c_l$	left neighboring cell of cell c
$c.c_r$	right neighboring cell of cell c
$c.2nddb$	2nd *DB* distance of cell c
W_{min}	minimum cell width

Algorithm 2 2nd DB-aware Leakage Optimization.

```
Procedure: $ReduceLeak()$
Input: a netlist $D$, a placement of $D$
Output: an optimized netlist
1: $Relocation(D)$
2: $M \leftarrow \varnothing$ ; $k \leftarrow 0$
3: for all cell $c$ in the netlist $D$ do
4:     if cell $c$ is not a flip-flop then
5:         if cell $c$ is downsizable then
6:             $m_k.c \leftarrow c$ ; $m_k.m \leftarrow downsize$
7:             update $c.c_l.2nddb$, $c.c_r.2nddb$
8:             $m_k.sf \leftarrow \Delta leakage/\Delta slack$ ; $M \leftarrow M \cup m_k$ ; $k \leftarrow k+1$
9:         end if
10:        if cell $c$ is not a highest $Vt$ then
11:            $m_k.c \leftarrow c$ ; $m_k.m \leftarrow downVt$
12:            $m_k.sf \leftarrow \Delta leakage/\Delta slack$ ; $M \leftarrow M \cup m_k$ ; $k \leftarrow k+1$
13:        end if
14:        if cell $c.c_l$ or $c.c_r$ is a different DB type from cell $c$ then
15:            $m_k.c \leftarrow c$ ; $m_k.m \leftarrow changeDB$
16:            update $c.c_l.2nddb$, $c.c_r.2nddb$
17:            $m_k.sf \leftarrow \Delta leakage/\Delta slack$ ; $M \leftarrow M \cup m_k$ ; $k \leftarrow k+1$
18:        end if
19:    end if
20: end for
21: while $M \neq \varnothing$ do
22:     Pick $m_k$ with the maximum $m_k.sf$ in $M$
23:     Commit $m_k$ ; $M \leftarrow M \backslash m_k$
24:     if TNS degrades then
25:         Revert
26:     end if
27: end while
```

3.4 2^{nd} DB-Aware Sizing

Algorithm 2 describes our sensitivity-based methodology for gate sizing, Vt swapping and DB swapping. Similar to the methodology in Section 3.3, we greedily perform either gate sizing, Vt swapping or DB swapping for each cell according to a pre-sorted list. Line 2 initializes an empty set of candidate moves. In Lines 3–20, we first populate the list with all candidate moves per cell, i.e., swapping to a higher Vt, downsizing and DB swapping. $\Delta leakage/\Delta slack$ is used as our sensitivity function and $\Delta slack$ denotes the timing slack difference for the swapped cell. Here, we only consider swaps of combinational cells, as shown in Line 4. In Lines 5–9, we add a candidate and perform the sensitivity calculations for downsizing if the cell is downsizable. Similarly, Lines 10–13 and Lines 14–17 add a candidate and calculate the sensitivity for Vt swapping and DB swapping, respectively. 2^{nd} DB distances are calculated for downsizing and DB swapping, as shown in Lines 7 and 16. In Lines 21–27, we then greedily commit the candidate move with the highest sensitivity as long as there is no TNS degradation.

4 EXPERIMENTS

4.1 Experimental Setup

We implement our heuristics in C++ with OpenAccess 2.2.43 [24] to support LEF/DEF [23]. We perform our experiments in a commercial 14*nm* FinFET technology with 9-track triple-Vt libraries. We apply our optimization to four design blocks, AES, MPEG, JPEG and VGA, from OpenCores [25]. We use two types of designs: Type-I has only SDB cells and Type-II has both SDB and DDB cells. We vary the target clock period from 0.3 to 0.5ns, with a step size of 0.05ns; and we vary the block utilization from 60% to 80%, with a step size of 5%. For each data point, we denoise the experiments using three runs, by varying the target clock period by ±1ps. We report the median value of the leakage recovery. We use *Synopsys Design Compiler L-2016.03-SP4* [27] for synthesis and *Cadence Innovus Implementation System v17.1* [22] for P&R, with leakage optimization in the highest effort mode. All the timing results in this section are reported by OpenSTA [26]. Due to different timing results between the commercial P&R tool and OpenSTA, we use a correlation methodology from [16] to correlate the P&R tool to OpenSTA. All experiments are performed with eight threads on a 2.6GHz Intel Xeon server.

We measure how much recovery we achieve by our flow, as shown in Equation (1).

$$Recovery = 1 - \frac{l_{opt} - l_{base}}{l_{actual} - l_{base}} \tag{1}$$

We use l_{base} to denote the leakage value reported by the P&R tool at post-P&R stage, which is 2^{nd} DB-oblivious. We use l_{actual} to denote the 2^{nd} DB-aware leakage value at post-P&R stage. We use l_{opt} to denote the leakage value after our optimization. Thus, $Recovery$ is the leakage recovery. Note that according to the metric $Recovery$, it is possible to have >100% leakage recovery. For example, if the 2^{nd} DB-aware analysis increases leakage from 0.182mW (l_{base}) to 0.190mW (l_{actual}), and our optimization reduces leakage to 0.180mW (l_{opt}), then we say that $Recovery$ is 125%.

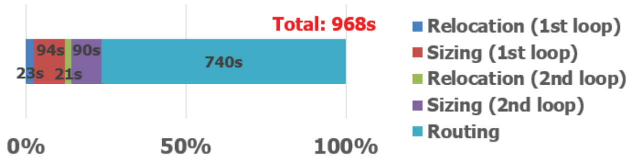

Figure 6: Runtime analysis for AES design. We use Type-II, 60% initial utilization and 0.5ns clock period. 1^{st} *loop* denotes the first iteration of our optimization and 2^{nd} *loop* denotes the second iteration. Sizing includes sensitivity calculations and cell moves for gate sizing, Vt swapping and DB swapping.

4.2 Design Space Exploration

In this section, we explore the sensitivity of results to design space choices. We explore three knobs for the experimental setup: (i) 2CPP padding for all flip-flops, (ii) DB swapping and (iii) multiple optimization iterations. We apply a minimum space of padding between flip-flops, because our optimization does not touch flip-flops. Flip-flops can be abutted with other cells but we require at least 2CPP space between flip-flops. We perform four sets of experiments to validate our experimental setup:

- **Baseline:** A design with 2CPP padding for flip-flops.
- **Expt. 1:** A design without 2CPP padding for flip-flops.
- **Expt. 2:** A design without DB swapping.
- **Expt. 3:** A design with four iterations ($i_{max} = 4$).
- **Expt. 4:** Sensitivity to timer setting (tolerance).

For the baseline, we use 2CPP padding for flip-flops and execute two iterations including DB swapping. Flip-flops are wide and they contribute a large portion of the 2^{nd} DB leakage increase to neighboring cells. Expt. 1 shows that leakage recovery is degraded by 29.7% without 2CPP padding for flip-flops. In Expt. 2, we compare our baseline to the optimization with DB swapping and show that 2.2% leakage recovery can be achieved with the addition of DB swapping. In Expt. 3, we perform our optimization in four iterations and compare the result to the baseline, where we only have two iterations of optimization. Four iterations give more leakage recovery in exchange for longer runtime. In Expt. 4, we explore the sensitivity to timer tolerance. The tolerance for the timer denotes a minimum percentage change in delay that causes propagated delays to be recomputed during incremental timing updates. We sweep the tolerance from 0% to 1% with a 0.1% step. As shown in Table 5, a design with 0.8% tolerance shows 67% runtime improvement and the same leakage recovery compared to a design with 0% tolerance. As a result, we choose 2CPP padding for flip-flops as our default P&R setup, two iterations of optimization including DB swapping, and 0.8% tolerance in incremental timing updates for all reported results in Section 4.3.

4.3 Main Results

We conduct two overarching experiments to show results of our flow. The first experiment shows the leakage optimization results over four design blocks. The second experiment further studies the sensitivity of leakage recovery to target clock period and initial utilization of the design.

Table 3: Experimental results for all the design blocks. We set 60% initial utilization and 0.5ns target clock period. The table reports (i) the number of instances in a design (#Inst.), (ii) design type which defined in Figure 1 (Type), (iii) worst setup slack (WS), (iv) total negative setup slack (TNS), (v) percentage of leakage increase from 2^{nd} DB-unaware analysis (Δ%), (vi) runtime of leakage optimization including relocation and sizing (LeakOpt), and (vii) runtime of incremental routing by a commercial P&R tool (Routing).

Design information			2^{nd} DB-unaware			2^{nd} DB-aware			Our results				Runtime		
Design	#Inst.	Type	WS (ns)	TNS (ns)	l_{base} (mW)	WS (ns)	TNS (ns)	l_{actual} (Δ%) (mW)	WS (ns)	TNS (ns)	l_{opt} (Δ%) (mW)	*Recovery*	LeakOpt (s)	Routing (s)	Total (s)
AES	13806	Type-I	0.011	0.000	0.225	0.012	0.000	0.295 (31%)	0.001	0.000	0.227 (1%)	97.1%	207	819	1026
	13279	Type-II	0.002	0.000	0.182	0.002	0.000	0.190 (4%)	-0.001	-0.002	0.180 (-1%)	>100%	228	740	968
MPEG	12380	Type-I	0.004	0.000	0.227	0.005	0.000	0.272 (20%)	-0.001	-0.003	0.245 (8%)	60.0%	257	902	1159
	12250	Type-II	0.004	0.000	0.228	0.004	0.000	0.264 (16%)	0.002	0.000	0.237 (4%)	75.0%	261	872	1133
JPEG	47061	Type-I	0.002	0.000	0.683	0.002	0.000	0.867 (27%)	0.001	0.000	0.745 (9%)	66.3%	1820	6911	8731
	39422	Type-II	0.003	0.000	0.697	0.003	0.000	0.765 (10%)	0.002	0.000	0.714 (2%)	75.0%	2039	7093	9132
VGA	67491	Type-I	0.011	0.000	1.203	0.012	0.000	1.786 (48%)	0.001	0.000	1.295 (8%)	84.2%	3819	29183	33002
	67546	Type-II	0.015	0.000	1.230	0.015	0.000	1.449 (18%)	-0.002	-0.010	1.256 (2%)	88.1%	3998	30799	34797

Leakage optimization. The design information and experimental results are summarized in Table 3. We use four design blocks, AES, MPEG, JPEG and VGA, with both Type-I and Type-II designs. We show worst slack (WS), total negative slack (TNS), leakage (Leak), leakage recovery (*Recovery*) and runtime. Going from 2^{nd} DB-unaware to 2^{nd} DB-aware analysis, the leakage is increased by up to 43%. After our leakage optimization flow with incremental routing and timing recovery performed by the commercial tool, we achieve 80% leakage recovery on average, with negligible timing degradation. Figure 6 shows the runtime breakdown in our flow. We can see that 24% of runtime is spent on gate sizing, Vt swapping and DB swapping, including sensitivity calculations. 76% of runtime is spent on the incremental routing by a commercial P&R tool. Next, we verify the robustness of our optimization with different clock period and utilization.

Sensitivity to target clock period and utilization. Leakage increases due to the 2^{nd} DB effect and optimization results vary by initial utilization and target clock period. We sweep initial utilization and target clock period for each design so as to study design impacts and our achievement. Figures 7(a) and (b) show leakage increments due to the 2^{nd} DB effect. Figures 7(c) and (d) show percentage recovery of the leakage after our optimization. In this experiment, 70% initial utilization is used for target clock period sweep, and 0.4ns clock period is used for initial utilization sweep. As target clock period decreases, the Type-I case has more leakage increments caused by the 2^{nd} DB effect because cells are closely placed. [3] However, there are smaller changes for the Type-II case because DDB cells mitigate the 2^{nd} DB effect. On the other hand, as initial utilization increases, both the Type-I and Type-II cases have more or similar effects. Our optimization with various clock period and utilization achieves 80% leakage recovery on average. For the testcases, the recovery seen in Figures 7(c) and (d) is stable across clock periods and utilizations.

5 CONCLUSIONS

In this work, we study the 2^{nd} DB effect, which is one of the critical LLEs for physical design in sub-10nm technologies, along with the mixed-DB design that can have both SDB and DDB cells. The 2^{nd} DB effect is a new challenge to the physical design flow, and designers must take this effect into account in order to mitigate model-hardware miscorrelation. We propose new heuristics to recover leakage power increments caused by the 2^{nd} DB effect, including the use of 2^{nd} DB-aware relocation, gate sizing, Vt swapping and DB swapping while satisfying placement constraints. Our work achieves 80% leakage recovery on average using the proposed flow. Our future work directions include: (i) detailed placement considering inter-row minimum implant width and spacing constraints due

[3]With a tight timing constraint, there are more large cells and cells are more closely placed to reduce the net delay, resulting in larger 2^{nd} effect.

Table 4: Experimental results for design space exploration. We use AES Type-II design with 60% initial utilization and 0.5ns target clock period.

Expt.	2^{nd} DB-unaware		2^{nd} DB-aware		Our results		
	TNS (ns)	l_{base} (mW)	TNS (ns)	l_{actual} (Δ%) (mW)	TNS (ns)	l_{opt} (Δ%) (mW)	*Recovery*
Baseline	0.000	0.182	0.000	0.190 (4%)	-0.002	0.180 (-1%)	>100%
Expt. 1	0.000	0.178	0.000	0.198 (11%)	-0.002	0.186 (4%)	60.0%
Expt. 2	0.000	0.182	0.000	0.190 (4%)	-0.002	0.183 (1%)	87.5%
Expt. 3	0.000	0.182	0.000	0.190 (4%)	-0.003	0.178 (-2%)	>100%

Table 5: Experimental results for sensitivity to the tolerance. We use AES Type-II design with 60% initial utilization and 0.5ns target clock period. The 2^{nd} DB-aware leakage power of the design is 0.190mW as shown in Table 3. The table reports (i) runtime of relocation step (Reloc.), (ii) runtime of sizing (Sizing), (iii) runtime of leakage optimization including relocation and sizing (LeakOpt), (iv) worst setup slack (WS), and (v) 2^{nd} DB-aware leakage power after optimization (Leak). 1^{st} loop denotes the first iteration of the optimization, and 2^{nd} loop denotes the second iteration.

Tolerance	1^{st} Loop		2^{nd} Loop		Our results		
	Reloc. (s)	Sizing (s)	Reloc. (s)	Sizing (s)	LeakOpt (s)	WS (ns)	Leak (mW)
0%	23	320	22	316	681	-0.001	0.180
0.1%	23	152	21	149	345	-0.001	0.180
0.2%	23	142	22	140	327	-0.001	0.180
0.3%	23	137	22	132	314	-0.001	0.180
0.4%	23	134	22	130	309	-0.001	0.180
0.5%	23	128	21	125	297	-0.001	0.180
0.6%	23	119	22	117	281	-0.001	0.180
0.7%	23	107	22	103	255	-0.001	0.180
0.8%	23	94	21	90	228	-0.001	0.180
0.9%	23	90	22	88	223	-0.001	0.184
1.0%	23	87	22	84	216	-0.001	0.187

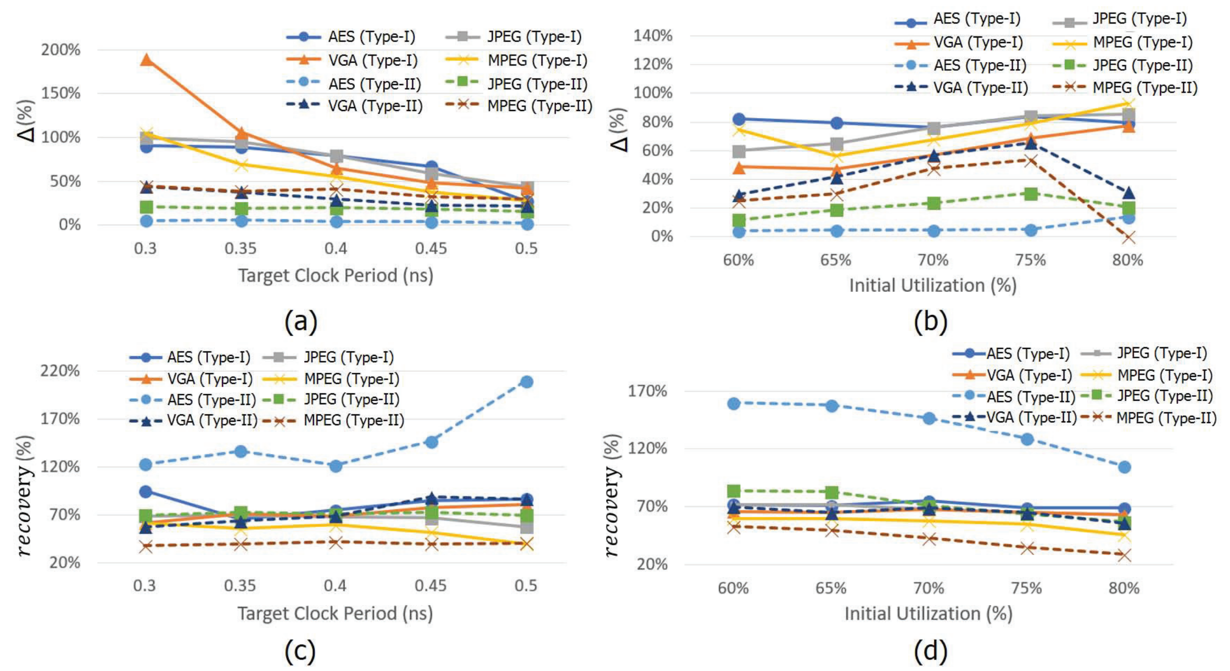

Figure 7: Studies of leakage power with varying target clock period and initial utilization. (a) Leakage increment from the 2^{nd} DB effect vs. target clock period. (b) Leakage increment from the 2^{nd} DB effect vs. initial utilization. (c) Leakage recovery by our optimization vs. target clock period. (d) Leakage recovery by our optimization vs. initial utilization.

to the mix of SDB and DDB cells, (ii) more comprehensive study of sensitivity functions for improved leakage recovery considering multiple corners, and (iii) support of mixed-DB within one cell, e.g., PMOS has SDB and NMOS has DDB within one standard cell.

ACKNOWLEDGMENTS

Research at UCSD is supported by Qualcomm, Samsung, NXP Semiconductors, Mentor Graphics, DARPA (HR0011-18-2-0032), NSF (CCF-1564302) and the C-DEN center.

REFERENCES

[1] C. Auth, A. Aliyarukunju, M. Asoro, D. Bergstrom, V. Bhagwat et al., "A 10nm High Performance and Low-Power CMOS Technology Featuring 3^{rd} Generation FinFET Transistors, Self-Aligned Quad Patterning, Contact Over Active Gate and Cobalt Local Interconnects", *IEEE IEDM*, 2017, pp. 29.1.1-29.1.4.

[2] R. Berthelon, F. Andrieu, E. Josse, R. Bingert, O. Weber, E. Serret et al., "Design / Technology Co-optimization of Strain-induced Layout Effects in 14nm UTBB-FDSOI CMOS: Enablement and Assessment of Continuous-RX Designs", *Proc. VTS*, 2016, pp. 1–2.

[3] Y. Du and M. D. F. Wong, "Optimization of Standard Cell Based Detailed Placement for 16nm FinFET Process", *Proc. DATE*, 2014, pp. 1–6.

[4] G. Flach, T. Reimann, G. Posser, M. Johann and R. Reis, "Effective Method for Simultaneous Gate Sizing and Vth Assignment Using Lagrangian Relaxation", *IEEE Trans. on CAD* 33(4) (2014), pp. 546–557.

[5] P. Gupta, A. B. Kahng, P. Sharma and D. Sylvester, "Gate-Length Biasing for Runtime Leakage Control", *IEEE Trans. on CAD* 25(8) (2006), pp. 1475–1485.

[6] N. Ha, Y.-D. Kim, B.-H, Lee, H.-O. Kim, K. Jeong and J. Kim, "System and Method of Designing Integrated Circuit by Considering Local Layout Effect", *US Patent Appl.*, US20180032658A1, 2018.

[7] K. Han, A. B. Kahng and H. Lee, "Scalable Detailed Placement Legalization for Complex Sub-14nm Constraints", *Proc. ICCAD*, 2015, pp. 867–873.

[8] C. Han, K. Han, A. B. Kahng, H. Lee, L. Wang and B. Xu, "Optimal Multi-Row Detailed Placement for Yield and Model-Hardware Correlation Improvements in Sub-10nm VLSI", *Proc. ICCAD*, 2017, pp. 667–674.

[9] C. Han, A. B. Kahng, L. Wang and B. Xu, "Enhanced Optimal Multi-Row Detailed Placement for Neighbor Diffusion Effect Mitigation in Sub-10nm VLSI", *IEEE Trans. on CAD* (2018), DOI:10.1109/TCAD.2018.2859266.

[10] J. Hu, A. B. Kahng, S. Kang, M.-C. Kim and I. L. Markov, "Sensitivity-guided Metaheuristics for Accurate Discrete Gate Sizing", *Proc. ICCAD*, 2012, pp. 233–239.

[11] A. B. Kahng, S. Kang, H. Lee, I. L. Markov and P. Thapar, "High-Performance Gate Sizing with a Signoff Timer", *Proc. ICCAD*, 2013, pp. 450–457.

[12] A. B. Kahng and H. Lee, "Minimum Implant Area-Aware Gate Sizing and Placement", *Proc. GLSVLSI*, 2014, pp. 57–62.

[13] A. B. Kahng, J. Dodrill and P. Gupta, "Designing in Advanced Technologies: A Quick Review of Approaches", Tutorial 4, *DAC*, 2018.

[14] T. Luo, D. Newmark and D. Z. Pan, "Total Power Optimization Combining Placement, Sizing and Multi-Vt Through Slack Distribution Management", *Proc. ASP-DAC*, 2008, pp. 352–357.

[15] W. K. Mak, W. S. Kuo, S. H. Zhang, S. I. Lei and C. Chu, "Minimum Implant Area-Aware Placement and Threshold Voltage Refinement", *IEEE Trans. on CAD* 36(7) (2017), pp. 1103–1112.

[16] C. W. Moon, P. Gupta, P. J. Donehue and A. B. Kahng, "Method of Designing a Digital Circuit by Correlating Different Static Timing Analyzers", *US Patent*, US7823098B1, 2010.

[17] L. Wei, K. Roy and C.-K. Koh, "Power Minimization by Simultaneous Dual-Vth Assignment and Gate-Sizing", *Proc. CICC*, 2000, pp. 413–416.

[18] R. Xie, K.-Y. Lim, M. G. Sung and R. R.-H. Kim, "Methods of Forming Single and Double Diffusion Breaks on Integrated Circuit Products Comprised of FinFET Devices and the Resulting Products", *US Patent*, US9412616, 2016.

[19] S. Yang, Y. Liu, M. Cai, J. Bao, P. Feng, X. Chen et al., "10nm High Performance Mobile SoC Design and Technology Co-developed for Performance, Power, and Area Scaling", *Proc. VTS*, 2017, pp. T70–T71.

[20] P. Zhao, S. M. Pandey, E. Banghart, X. He, R. Asra, V. Mahajan et al., "Influence of Stress Induced CT Local Layout Effect (LLE) on 14nm FinFET", *Proc. VTS*, 2017, pp. T228–T229.

[21] Design Technology Team, Samsung Electronics Co. Ltd., *Personal Communication*, May 2018.

[22] Cadence Innovus User Guide, http://www.cadence.com

[23] LEF/DEF Reference 5.7. http://www.si2.org/openeda.si2.org/projects/lefdef

[24] Si2 OpenAccess. http://www.si2.org/?page=69

[25] OpenCores: Open Source IP-Cores, http://www.opencores.org

[26] OpenSTA: Static Timing Analyzer, https://github.com/abk-openroad/OpenSTA

[27] Synopsys Design Compiler User Guide, http://www.synopsys.com