Heuristic Methods for Fine-Grain Exploitation of FDSOI

Hamed Fatemi, Andrew B. Kahng, *Fellow, IEEE*, Hyein Lee, and José Pineda de Gyvez, *Fellow, IEEE*

***Abstract*—Fully depleted silicon on insulator (FDSOI) is attractive for its low cost and low power; the mixed-Vt and body-bias levers that it affords expand the performance-power solution space. However, in FDSOI, different-Vt (i.e., low Vt and regular Vt) devices must be isolated from each other, which makes realization of fine-grained mixed-Vt/body-biasing in layout extremely challenging. In this article, we study heuristic methods aimed at exploitation of fine-grained mixed-Vt in FDSOI implementation. We propose a novel speed domain partitioning (SDP) problem formulation that comprehends the spatial contiguity restrictions arising from flip-well structure of low Vt regions in popular 28-nm commercial FDSOI offerings. We explore a wide space of implementation flows that include an integer linear programming (ILP)-based approach, and a heuristic (sensitivity-based) optimization. Our experimental studies have been performed across multiple commercial enablements. We observe that outcomes are library- and design-dependent. For implementations using generic library options, up to 20% speed improvement with 54% low Vt region is seen for one out of four testcases studied. For implementations using "rich" library options, up to 7% speed improvement with 26% low Vt region is achieved. We provide a discussion that summarizes root-cause, intrinsic difficulties of fine-grained exploitation of mixed-Vt. Finally, we suggest a "decision tree" to help assess a design's amenability to fine-grained mixed-Vt implementation, and to help guide design flow selection for better design QoR.**

***Index Terms*—Body-bias, co-optimization, fully depleted silicon on insulator (FDSOI), mixed-Vt, placement, sizing.**

I. Introduction

FULLY depleted silicon on insulator (FDSOI) is a promising process technology especially for low power Internet of Things designs due to its low-cost and low-power potential. Particularly, contrary to the FinFET process in which body bias is inefficient, body bias in FDSOI is a good knob for speed and leakage optimization. At a 0.5-V supply, speed can be improved by up to 5.5× by using forward body bias (FBB), and leakage power can be reduced by up to 50× by using reversed body bias (RBB) [6].

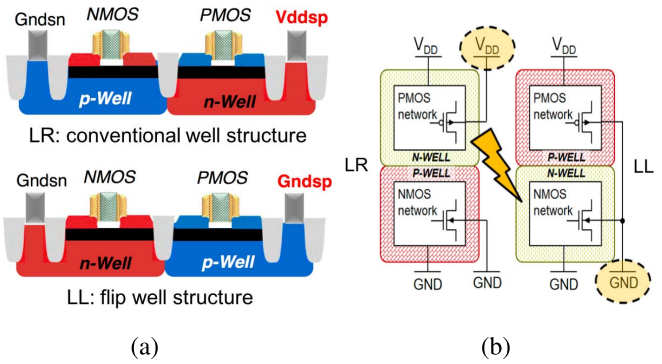


Fig. 1. (a) FDSOI device structure. (b) LL and LR cells cannot be abutted due to well bias conflicts [6].

Due to the unique FDSOI device structure that does not have body, regular Vt (LR) and low Vt (LL) devices are implemented by a special structure called a *flip-well* configuration.[1] Fig. 1 shows the structures of a conventional well (LR) and a flip well (LL). In the flip-well structure, an *N*-well is implemented under the nMOS transistor, and a *P*-well is implemented under the pMOS transistor; this structure enables a wide range of FBB [8]. However, as the wells are flipped, abutting LL and LR cells induces a well bias conflict. Thus, LL and LR cells must be isolated from each other.[2]

Successful implementations of mixed-Vt and body bias in FDSOI are well-documented in several previous works [3], [6], [8]. In these works, the Vt and body bias are assigned in a *coarse-grained*, i.e., block-level, manner due to the constraint that different-Vt cells cannot be abutted. To complement these previous works, and to ensure that the Vt and body bias options in FDSOI are fully exploited, *fine-grained* implementations should also be considered and

Manuscript received December 13, 2018; revised May 26, 2019; accepted July 26, 2019. Date of publication August 13, 2019; date of current version September 18, 2020. This article was recommended by Associate Editor L. Behjat. *(Corresponding author: Hyein Lee.)*

H. Fatemi is with NXP Semiconductors, San Jose, CA 95134 USA (e-mail: hamed.fatemi@nxp.com).

A. B. Kahng is with the Department of Computer Science and Engineering, University of California at San Diego, San Diego, CA 92093 USA, and also with the Department of Electrical and Computer Engineering, University of California at San Diego, San Diego, CA 92093 USA (e-mail: abk@ucsd.edu).

H. Lee was with the Department of Electrical and Computer Engineering, University of California at San Diego, San Diego, CA 92093 USA. She is now with Synopsys Inc., Sunnyvale, CA 94085 USA (e-mail: hyein.lee1@synopsys.com).

J. Pineda de Gyvez is with NXP Semiconductors, Eindhoven, The Netherlands, and also with NXP Semiconductors, San Jose, CA 95134 USA (e-mail: jose.pineda.de.gyvez@nxp.com).

Digital Object Identifier 10.1109/TCAD.2019.2935053

[1]The names LL and LR map to nomenclature, such as LVT/SLVT or HVT/RVT used in popular foundry technologies. In this article, we generically use LL/LR to avoid the use of any foundry-specific names.

[2]There exist some foundry technologies which offer different Vt with the same well structure (e.g., 22FDX from *Global Foundries* [19]), enabling different-Vt devices to be mixed freely. However, in this article, we refer to "mixing different well structures" as mixed-Vt: we study the regime where different Vts are generated by using different well structures. Specifically, LL (resp. LR) is formed by a flip (resp. conventional) well. This corresponds to widely used 28-nm commercial offerings.

evaluated. However, it is not straightforward to estimate the benefits of fine-grained mixed-Vt and body bias implementation due to the placement constraints for different-Vt options. Indeed, our present work shows that in the FDSOI context, benefits realized from fine-grained use of Vt (hence, body bias) options appear strongly dependent on designs, libraries, performance targets and power requirements. For example, our results and discussion below suggest that realizable benefits depend on: 1) availability of rich cell library options that offer power-delay tradeoff, such as poly biasing options; 2) the optimized timing structure produced by traditional physical implementations, including aspects, such as multiplicity of timing-critical paths ("wall of slack") and the spatial distribution of critical instances; and 3) a given design's sensitivity to active leakage and dynamic power.

In this article, we study the potential of *fine-grained*, i.e., sub-block-level, mixed-Vt assignment in FDSOI. We frame this article using a novel speed domain partitioning (SDP) problem formulation, since the Vt assignment essentially seeks to partition the input block into fast and slow parts. Note that in the following, we focus on the challenge of fine-grained mixed-Vt assignment in FDSOI, and we discuss only briefly in Section V the (more difficult) challenge of fine-grained body bias assignment. Both of these challenges share the fundamental problem of determining rectilinear layout regions for speed boost, i.e., performance improvement, with minimum power increase. Moreover, the mixed-Vt assignment problem that we study can be seen as closely related to body bias assignment—e.g., FBB assignment can be comprehended by adding more timing constraints for the FBB mode.

Our main contributions are summarized as follows.

1) We formulate the SDP problem and develop two basic optimization flows to address the SDP problem: an integer linear programming (ILP)-based flow, and a sensitivity function (SF)-based heuristic flow.
2) We experimentally study the potential benefits of fine-grained mixed-Vt in FDSOI. Up to 20% (resp. 7%) speed improvement with 54% (resp. 26%) LL region area is achieved for an example with generic (resp. "rich") cell library. This said, we observe that outcomes are highly dependent on available library cells as well as characteristics of the input designs.
3) We analyze root-cause challenges to fine-grained mixed-Vt exploitation in FDSOI. Specifically, we identify three intrinsic difficulties: a) availability of rich library cell options; b) the existence of a "slack wall" in well-optimized designs; and c) spatial contiguity constraints on the placement.
4) We suggest a *decision tree* to help assess the potential benefits for a given design of using mixed-Vt in FDSOI.
5) Finally, we briefly explain fundamental difficulties of post-placement fine-grained body-biasing that we have encountered in our experimental investigations. Future research can potentially revisit these challenges by exploring improved implementation flows for fine-grained body-biasing, e.g., that apply preplacement netlist optimizations and useful skew.

The remainder of this article is organized as follows. Section II reviews related works. Section III describes our approaches for island generation. In Section IV, we describe our experimental setup and results. In Section V, based on our experimental results, we give analyses of inherent difficulties of exploiting fine-grained mixed-Vt in FDSOI. Section VI gives conclusions along with a brief discussion on the difficulties of fine-grained body-biasing in FDSOI, and directions for future work.

II. Related Works

We now review related works. To the best of our knowledge, there are not many works that directly address fine-grained mixed-Vt and body biasing in FDSOI. Thus, along with fine-grained body bias work, we review previous works in multiple-Vdd (voltage island) placement as well as multiple-height cell placement. The three problems have similarity in that cells are assigned to certain attributes, with consideration of placement constraints, to optimize design speed and power.

A. Island Generation for Dual Vdds

A post-placement Vdd assignment flow to minimize the number of level converters is proposed in [7]. Sensitivity-based Vdd assignment is followed by placement optimization based on soft clustering using a min-cut placer to generate voltage islands with a reduced number of level converters. Liu *et al.* [10] proposed a voltage island generation method in placement for dual-Vdd designs. The proposed flow starts with power- and timing-driven placement. Sensitivity-based voltage assignment is performed followed by partition-based placement refinement with soft clustering of the same Vdd cells. During the placement refinement stage, neighboring bins are merged to create a new larger bin, and this new bin is repartitioned considering wirelength and clustering for voltage isolation. This process is performed iteratively until all the same-Vdd cells are clustered.

B. Island Generation for Multiple Vdds

Wu *et al.* [13] proposed a dynamic programming-based methodology to group voltage islands for designs with multiple supply voltages. The voltage island grouping problem is formulated as follows. Given a set of minimum Vdd assignment v_g for each grid g, and an error threshold δ, find a partitioning with the smallest size (i.e., number of islands) where each island has an error of at most δ. The error of an island I is defined as $\sum_{g \in I} (v_{\max} - v_g)$, where $v_{\max} = \max_{g \in I} v_g$. The heuristic method in [13] has two steps: 1) size reduction to a $p \times q$ array G where each grid has an error less than δ so that the array is manageable by dynamic programming-based approach and 2) applying dynamic programming to G. In the work of [11], a greedy heuristic approach for rectangular voltage island generation is proposed for multiple-Vdd designs. The largest rectangular regions with the minimum resulting power are selected iteratively for a given placement along with grid-based voltage assignment.

C. Row-Based Dual Vdds

Yeh *et al.* [16] proposed cell layout techniques along with a simulated annealing-based placement algorithm that support row-based dual-Vdd designs. Xiang *et al.* [14] proposed an improved placement algorithm that handles local clock buffers for the row-based dual-Vdd designs. The proposed flow consists of two stages: 1) clustering gates to form voltage islands and 2) linear programming-based legalization. In this article, latches are grouped based on maximum weighted matching. Then, gates are again clustered based on their distance-based weights, followed by min-cost max flow-based level shifter assignment. Xiang *et al.* [15] proposed an extension of [14] that improves timing by moving gates for the row-based dual-Vdd designs. The proposed flow moves timing-critical gates to feasible locations in a greedy manner, without changing voltage assignments.

D. Mixed-Height Cell Placements

The work of [5] proposes a placement optimization flow to implement a fine-grained noninteger multiple-height cell placement. dynamic programming-based partitioning is followed by sensitivity-based gate sizing and placement optimization in the proposed flow.

E. Other Fine-Grained Body Bias Work

Flores [2] proposed a greedy algorithm that determines a body bias island floorplan that gives minimum wirelength. In the proposed algorithm, the size and location of the islands are selected based on track utilization. Taco *et al.* [12] studied gate-level dynamic body biasing in 28-nm FDSOI in the circuit level. Taco *et al.* [12] compared the dynamic threshold voltage metal-oxide semiconductor field-effect transistor (MOSFET) circuit (transistor-level body biasing) and the gate-level body-biased circuit [1], [4]. Kühn *et al.* [9] proposed a body bias domain partitioning method by identifying gates activated through a common identifier during logic synthesis. Kühn *et al.* [9] assign body bias to partitioned domains based on leakage and timing.

In sum, previous works have addressed design optimizations in contexts—notably, voltage island and mixed-height placement—that are similar to our present FDSOI context. Many of these other works report noteworthy power/speed/area benefits from their proposed optimizations. However, as seen from our experimental results as well as Section V discussion below, the SDP problem in FDSOI seems fundamentally more challenging for several reasons.

III. Problem Statement and Our Approaches

In this section, we first formulate the SDP problem for mixed-Vt FDSOI implementation. We then describe a set of implementation approaches we have tried, and give details of the two best approaches that empirically give maximum speed improvements (for given power overhead) subject to placement constraints.

It must be emphasized that there are many conceivable ways to implement fine-grained mixed-Vt (and body bias) implementations, working at various design levels in the RTL-to-GDS flow. For example, one could plausibly synthesize with both Vts and partition the netlist according to the Vt of each gate. Or, one could synthesize with LR-only and optimize the netlist while making LL cells additionally available. Among many possible implementation flows that we have investigated, our discussion focuses on *post-placement* optimizations in which we start from placed netlists *implemented with LR-only cells*, without awareness of placement constraints in FDSOI. We have zeroed in on this space of implementation flows, for the following reasons. First, it is difficult to identify timing-critical cells that will eventually require speed boost, based on a preplacement netlist—since the timing changes disruptively after placement. Second, if we predetermine (LL) regions for speed boost, this restricts placement and leads to poor quality of placement results with respect to timing and wirelength. Third, if we allow the mixing of LL/LR Vt values up front in synthesis, the optimization of LL versus LR regions becomes highly restricted by existing (placements of) LL cells; in our experience, this leads to very large timing and/or power penalties.

Mixed-Vt Speed Domain Partitioning (SDP-MVT) Problem: Given an initial placed design implemented with LR cells only, perform Vt swapping, sizing, and placement optimization to define LL regions under timing/placement constraints.

Input: A placed design, synthesized, and optimized with LR cells.

Output: An optimized mixed-Vt netlist/placement, with LL islands.

Constraints: The optimization would typically be subject to timing and placement constraints. For example, the target operating frequency of the mixed-Vt implementation should be $X\%$ higher than that of the pure-LR implementation; the total power of the mixed-Vt implementation should be no more than $Y\%$ higher than that of the pure-LR implementation; and the total area of LL regions should be no more than $Z\%$ of the total area of the design. Our experiments below attempt to shed light on achievable combinations of X, Y, and Z for the testcases studied.

A. Overall Flow

Fig. 2 illustrates our overall flow. The input to the overall flow is a placement implemented and optimized with LR cells only. We generate LL islands and assign LL to all the cells in the islands. We then further optimize timing if needed without changing the LL assignment. The output is an optimized placement (and netlist) with LR and LL cells. Our background experiments have tried a rather large number (i.e., $4 \times 5 = 20$) of flow variants with various combinations of island generation methods and timing optimization methods. Additionally, various alternative ILP-based assignment strategies have been tested.

For the island generation, we have tried four flows, namely, (A) a "brute-force" approach [2]; (B) sensitivity-based assignment; (C) ILP-based assignment; and (D) iterative heuristic-based assignment. In the brute-force approach, we first collect all the timing-critical cells in a given placement, and move

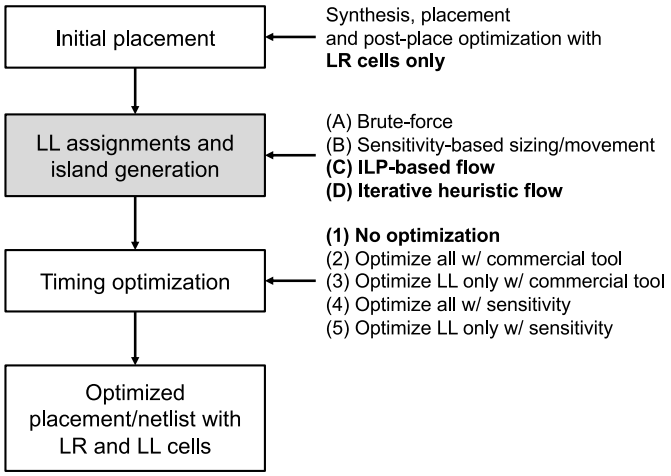


Fig. 2. Overall flow, showing options explored in background studies. Superior flow options are shown in bold font.

these cells into a predefined rectangular region. We sweep the locations and aspect ratios of the predefined region to find a best solution; thus, the brute-force method is similar to executing the method of [2] with many different target regions. The sensitivity-based assignment is performed as follows, with respect to a coarse gridding of the layout region into ($\sim$100) grids. First, for each cell in the layout, we calculate the sensitivity (1/Δ total negative slack) when the LR cell is changed to LL. We then select the top-*k*% of all cells with respect to the sensitivity. Among the selected cells, we randomly choose *n* cells to use as seeds for the LL region generation. (Since LL regions must be contiguous in the final layout solution, succeeding LL cells should be close to these seed LL cells.) Using the chosen *n* cells as "anchors," we recalculate the sensitivity of each cell considering proximity, i.e., the quotient (distance to the nearest anchor/Δ total negative slack). We then sum all of the cell sensitivity values in each grid. Finally, we assign LL to the regions (grids) with the largest sums of sensitivity values.

In our experiments, Flows (A) and (B) in Fig. 2 do not offer noticeable speed improvement. For Flow (A), we observe that forcing timing-critical cells to be placed in predefined LL (fast) regions is too disruptive to timing-driven placement. For example, critical cells could be placed far apart from their noncritical fanin/fanout cells, which results in excessive increase of wirelength (and wire capacitance). In such cases, new timing-critical paths occur in LR (slow) regions. Indeed, in our optimization results, none of the data points shows speed improvement as compared to the initial LR placements.

For Flow (B), although the LL regions are predefined with an understanding of the initial placement through sensitivity, the issue of creation of new timing-critical paths still exists. Timing-critical cells that are assigned to LL, but that are not in the LL regions, must be moved to nearest LL regions. This is again disruptive to the existing timing-/wirelength-driven placements. Also, since the sensitivity is calculated under the assumption that one cell is swapped while all the other cells are fixed, it does not show the whole picture. This is the motivation of our iterative heuristic-based assignment where we update timing after each step of gradual LL region assignment, so as to maintain more accurate timing.

TABLE I
TIMING OPTIMIZATION RESULTS

Flow	Clock Period	PLeak	PTot
(1)	1.000	1.482	9.94
(2)	0.969 (3%)	1.998 (35%)	11.00 (11%)
(3)	0.986 (1%)	2.010 (36%)	10.97 (10%)
(4)	0.998 (0%)	1.834 (24%)	10.70 (8%)
(5)	1.000 (0%)	1.482 (0%)	10.02 (1%)

For the timing optimization, we have tried five flows: (1) no optimization; (2) commercial tool-based optimization for the entire placement; (3) commercial tool-based optimization for islands where we run post-placement optimization for the LL regions only; (4) sensitivity-based gate sizing and movement for the entire placement; and (5) sensitivity-based gate sizing and movement for the LL regions only. For the sensitivity-based gate sizing 4) and 5), we estimate Δ slack for each potential move and swap. Δ slack/current slack is used as our SF.

Table I shows the results of the timing optimization flows applied to M0 designs obtained by the island generation Flow (D). The table reports the minimum achievable clock period, and the corresponding leakage and total power. No single timing optimization shows good power-delay tradeoff. For example, Flow 2) shows 35% and 11% more leakage and total power while only achieving 3% delay benefit, compared to Flow 1). This is because the (post-placement) timing optimization is limited to sizing-only due to the placement-induced constraint where Vt is fixed according to locations. The sizing-only optimization may be further limited by placement if only a limited number of empty sites are available for timing-critical cells that need upsizing.

From our experimental studies, we have concluded that ILP-based assignment and iterative heuristic-based assignment offer clearly superior results in terms of speed improvement for given placement constraints and power overhead. For timing optimization, none of the methods studied is helpful to improve worst setup slack. In sum, the best overall flows that we select for our experiments below are designated in bold font in Fig. 2.

B. Algorithms for Island Generation

We now give details of the two approaches for island generation that offer the best results in our experiments. In our island generation flows, we perform grid-based assignment, i.e., all the cells in the same grid have the same Vt. Thus, island generation first splits the layout into a number of uniform rectangular grids. Empirically, the results of our flows are not sensitive to the number of grids when this number is 100 or more. Thus, in the following we report results for 100 uniform rectangular grids.

1) ILP-Based Flow: In our ILP-based flow, we first collect timing critical paths and formulate timing constraints. Table II

TABLE II
NOTATIONS

Notation	Meaning
α	LL island area constraint (0 – 1)
N	maximum number of islands
i / j / m / n	index of cell / LL island / timing path / grid, respectively
v_n^j	binary indicator of whether grid n belongs to island j
v_j	binary indicator of whether island j is generated
g_i	grid index of cell i
Δd_i	delay difference between LR and LL for cell i LR cell delay subtracted by LL cell delay.
d_m	initial delay of path m
CP	clock period
max_area	maximum area constraint for islands
x_l^j, y_l^j (resp. x_u^j, y_u^j)	x, y locations of lower-left (resp. upper-right) corner of island j
x_l^n, y_l^n (resp. x_u^n, y_u^n)	x, y locations of lower-left (resp. upper-right) corner of grid n
x_l, y_l (resp. x_u, y_u)	x, y locations of lower-left (resp. upper-right) corner of the layout
G	large number

shows the notations used in our ILP formulation. The objective is to minimize the clock period CP [Constraint (1)]. CP is bounded by the maximum path delay in Constraint (5). Constraint (2) ensures that total area of islands is not more than a given maximum area. The area of a rectangle is estimated by its half-perimeter. Constraint (3) ensures that at most one island j is selected for a grid n. For each grid n, the sum of v_n^j must be equal or less than one. Constraint (4) ensures that the number of islands does not exceed N. The v_j is forced to be one if $v_n^j = 1$ for any n. When $\sum_n v_n^j = 0$, which means no single $v_n^j = 1$ exists, v_j is forced to be zero. The number of islands is counted by $\sum_j v_j$, and this must be less than or equal to N. Constraint (5) ensures the clock period CP to be larger than the new path delay that is computed considering the delay improvement of swapping to LL. Constraint (6) determines the dimension of islands such that every LL grid is covered by LL islands

$$\text{Minimize:} \quad \text{CP} \tag{1}$$

$$\text{Subject to:} \quad \sum_j \left(x_u^j - x_l^j + y_u^j - y_l^j \right) < \alpha \cdot (x_u - x_l + y_u - y_l) \tag{2}$$

$$\sum_j v_n^j \leq 1, \quad \forall n \tag{3}$$

$$v_j \geq v_n^j, \quad \forall n; \quad v_j \leq \sum_n v_n^j; \quad \sum_j v_j \leq N \tag{4}$$

$$d_m - \sum_{i \in \text{path}_m} \left(\sum_j v_{g_i}^j \right) \cdot \Delta d_i < \text{CP}, \quad \forall m \tag{5}$$

$$\begin{aligned} & x_u^j - G v_n^j + G > x_u^n, \quad \forall j \\ & x_l^j + G v_n^j - G < x_l^n, \quad \forall j \\ & y_u^j - G v_n^j + G > y_u^n, \quad \forall j \\ & y_l^j + G v_n^j - G < y_l^n, \quad \forall j. \end{aligned} \tag{6}$$

2) Iterative Heuristic Flow: Algorithm 1 shows our iterative, SF-based heuristic flow for LL island generation. We select LL island regions based on sensitivity, where the average slack of the region is used as the SF. The island regions are composed of contiguous grids. In line 1, we find a small rectangular region based on sensitivity, and use this as a seed for generating island regions. In lines 2–13, we grow the seed

Algorithm 1 SF-Based Heuristic Flow

```
Procedure: HeurVtAssign()
Input: placement, max_area
Output: placement with LL islands
1:  region ← FindBestRegion(area = 0.01, SF)
2:  while area < max_area do
3:    for direction in (east, west, north, south) do
4:      grow_region ← grow(region, direction)
5:      score ← CalcSF(grow_region, SF)
6:      if best_score < score then
7:        region ← grow_region
8:        best_score ← score
9:      end if
10:   end for
11:   area ← region.area
12:   update timing
13: end while
14: return region
```

gradually until the area of the region reaches a given max_area constraint. More specifically, for each direction (line 3), we try growing the seed region by one grid (line 4), and calculate SF (line 5). We select the best direction to grow, which gives the best (minimum) SF score (line 7).

IV. EXPERIMENTAL RESULTS

In this section, we report the results of our two best flows that we describe in Section III. For each of our testcases, we perform three distinct optimizations to find rectilinear regions for the mixed-Vt problem: 1) ILP with one island (ILP-1); 2) Iterative ILP with two islands (ILP-2); and 3) Heuristic with two islands (Heur). The Iterative ILP simply defines a first LL island based on the above-described ILP, and then—based on this LL assignment—sets up and solves the same ILP with updated timing information to define a second LL island. Our methods are implemented in Tcl 8.4 [23], and CPLEX v12.6.3 [20] is used as the ILP solver. Runtimes for each solution, including runtimes of commercial tool steps and any CPLEX runtimes, are at most 6 h on a 2.8-GHz Xeon server with 128-GB RAM.

A. Experimental Setup

We perform experiments in a 28-nm FDSOI foundry technology with dual-VT libraries, 0.9-V nominal supply voltage. We validate our flows with ARM Cortex M0 and M3 cores, along with two designs (ldpc, viterbi) from the OpenCores website [21].

The designs are synthesized with target periods in the range of 0.5–2.5 ns, with a 50-ps step. For each synthesized netlist, P&R is performed with three P&R target periods, i.e., synthesis target period $+\{-50, 0, 50\}$ ps. The SP&R (synthesis and P&R) is performed with three library options, i.e., LR-only, LL-only, and LR+LL (mixed-Vt), without placement constraints. For the P&R flow, high leakage power optimization effort is applied, along with post-placement leakage optimization. For each SP&R implementation, we record

TABLE III
TESTCASES

Enable	Name	#Instances	Min. LL period	Min. LR period
Enable1	M0	7K∼11K	0.803	1.102
	ldpc	46K∼59K	0.775	1.027
	M3	47K∼60K	1.151	1.577
	viterbi	53K∼69K	0.529	0.755
Enable2	M0-2	8∼13K	0.834	1.144
	M3-2	47∼71K	1.248	1.669

effective clock period (ECP), which we calculate as the target period subtracted by worst setup slack, along with leakage power and total power. (Assuming a simple single clock constraint, worst setup slack plus the target clock period is equal to the smallest clock period at which setup slack $= 0$.)

We have performed our experiments with multiple commercial 28-nm FDSOI enablements, one with 12-track (12T) cells and a "generic" (no poly bias) library, and the other with 8-track cells and a rich library with poly biases.[3]

1) *Enable1:* 28-nm 12T LL and LR without poly bias (i.e., P0) are used as library cells. Synopsys Design Compiler N-2017.09 [22] and Cadence Innovus v17.1 [17] are used for logic synthesis and P&R tools, respectively.
2) *Enable2:* 28-nm 8T LL and LR with four poly bias options (i.e., P0, P4, P10, and P16) are used as library cells. Cadence Genus v16.2 [18] and Cadence Innovus v15.2 [17] are used for logic synthesis and P&R tools, respectively.[4]

We have implemented the M0 and M3 testcases with each of these two enablements. We denote M0 (resp. M3) implemented with Enable2 as M0-2 (resp. M3-2).

Table III shows the testcase information. All the numbers are reported by the P&R tools that correspond to the associated enablements (i.e., Innovus v17.1 and v15.2 for Enable1 and Enable2, respectively), at the post-placement stage. The *Min. LL period* and *Min. LR period* columns show the minimum achievable effective period with LL-only and LR-only SP&R implementation, respectively. We observe that roughly 30%–40% speedup can be achieved with LL-only implementations (i.e., 100% LL), compared to LR-only implementations. This can be viewed as an upper bound on speedup that could be achieved by any mixed-Vt implementation.

B. Experimental Results

Recall that we perform three optimizations, i.e., ILP-1, ILP-2, and Heur, on LR-only implementations to find rectilinear regions for mixed-Vt FDSOI implementation. To obtain meaningful inputs to our optimization flows, we select several LR-only implementations with ECPs no more than $1.1\times$ *Min. LR period*. We run our optimizations with maximum LL% constraints of {20%, 30%, 40%, 50%}. We then report the results with minimum leakage power while meeting {2%, 5%, 7%, 10%, 20%} speed improvements compared to *Min. LR period*.

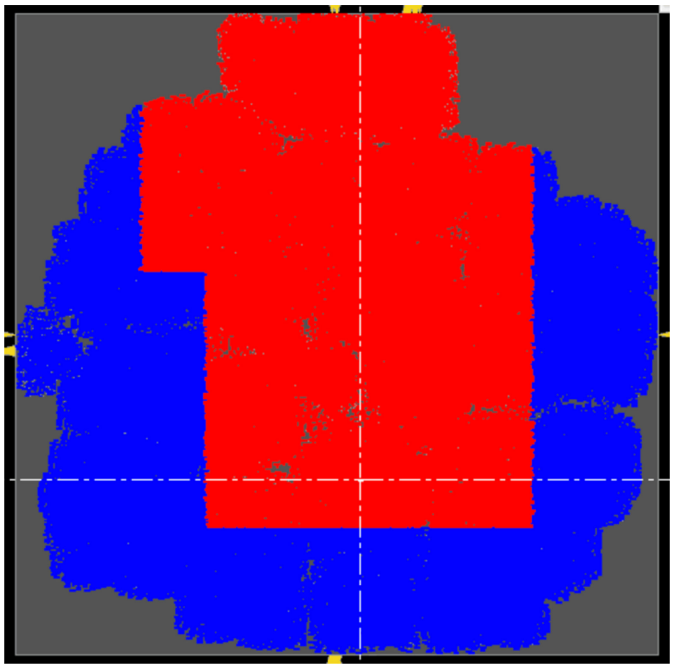


Fig. 3. Island shape obtained by ILP-2 for the viterbi testcase in Enable1. LL regions are highlighted in red.

Table IV shows the results of our three heuristics for mixed-Vt optimization (ILP-1, ILP-2, Heur), along with three reference implementations: 1) the LR-only (LR) baseline implementation; 2) the LL-only (LL) fastest-possible implementation; and 3) the mixed-Vt (without placement constraints) (LR+LL) implementation that bounds the mixed-Vt power-speed tradeoff. In each group of columns of the table, we report parameters [ECP, leakage (*PLeak*), and total power (*PTot*), and percentage area of LL region(s)] of the lowest-power solution that achieves the given percentage speed improvement, relative to the LR baseline implementation. We also report leakage and total power reduction in percentages as compared to LL in parenthesis in Columns PLeak and PTot. The best flow in terms of power values for each design is highlighted in bold font. Blanks (indicated by "-") mean that there is no result that meets the corresponding speed improvement with the corresponding flow. We note that our target periods are based on *Min. LR period* which are aggressive, and that the upper bound of the possible speedup is only 30%–40%. We also note that we do not include the implementation overhead due to the spacing rules between LL and LR. In reality, mixed-Vt implementations with placement constraints will have more area and thus consume larger total power.

Fig. 4 shows the leakage and total power versus the ECPs for the four designs implemented with Enable1, and for the two designs implemented with Enable2. Fig. 3 shows the island shape obtained by ILP-2 for the viterbi testcase in Enable1. This obtains a speedup of 20% with LL region area of 54%.

Our high-level findings are summarized as follows.

[3]Poly biasing refers to transistor gate (channel) length biasing, typically by a positive number of nanometers, for ultra fine-grain exploitation of leakage-delay tradeoff. For example, P10 denotes a +10 nm (relative to the nominal value) channel length.

[4] These particular tool versions were the only (latest) available tools for each enablement when the experiments were executed. The location of testcase data dictated the enablement used for given experiments.

TABLE IV
RESULTS OF ILP-1, ILP-2, AND HEUR

Design	Flow	2% imprv				5% imprv				7% imprv				10% imprv				20% imprv			
		ECP	PLeak	PTot	LL%	ECP	PLeak	PTot	LL%	ECP	PLeak	PTot	LL%	ECP	PLeak	PTot	LL%	ECP	PLeak	PTot	LL%
M0	LR	1.102	0.17	8.09	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	LL	1.059	1.15	7.42	100	1.008	1.24	7.96	100	1.008	1.24	7.96	100	0.996	1.31	8.57	100	0.917	1.49	9.59	100
	LR+LL	1.07	0.55(52%)	7.05(5%)	31	1.025	0.69(44%)	7.76(3%)	39	1.025	0.69(44%)	7.76(3%)	39	0.947	0.78(40%)	8.23(4%)	41	0.911	1.02(32%)	9.25(4%)	53
	ILP-1	1.053	**0.39(66%)**	8.14(-10%)	15	1.042	0.91(27%)	9.11(-14%)	30	1.007	**1.05(15%)**	9.28(-17%)	41	0.992	1.27(3%)	9.66(-13%)	52	0.915	1.51(-1%)	10.95(-14%)	53
	ILP-2	1.046	0.59(49%)	8.51(-15%)	19	1.046	0.59(52%)	8.51(-7%)	19	0.99	1.07(14%)	9.53(-20%)	40	0.99	**1.07(18%)**	9.53(-11%)	40	-	-	-	-
	Heur	1.045	0.45(61%)	8.25(-11%)	17	1.045	**0.45(64%)**	8.25(-4%)	17	1.027	1.13(9%)	9.47(-19%)	44	1	1.48(-13%)	10.02(-17%)	54	-	-	-	-
ldpc	LR	1.027	1.17	79.44	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	LL	0.972	7.58	89.37	100	0.972	7.58	89.37	100	0.941	8.49	91.21	100	0.916	8.75	86.74	100	0.82	12.41	108.42	100
	LR+LL	0.962	4.55(40%)	78.07(13%)	50	0.962	4.55(40%)	78.07(13%)	50	0.909	5.63(34%)	86.05(6%)	56	0.909	5.63(36%)	86.05(1%)	56	0.817	10.06(19%)	104.42(4%)	77
	ILP-1	1.003	5.51(27%)	85.49(4%)	22	0.977	11.31(-49%)	97.12(-9%)	53	-	-	-	-	-	-	-	-	-	-	-	-
	ILP-2	0.996	**4.14(45%)**	**84.53(5%)**	16	0.967	9.13(-20%)	93.2(-4%)	43	0.954	11.32(-33%)	102.89(-13%)	53	-	-	-	-	-	-	-	-
	Heur	1.004	7.24(4%)	88.39(1%)	40	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
M3	LR	1.577	0.75	29.87	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	LL	1.479	6.53	36.1	100	1.479	6.53	36.1	100	1.403	7.08	38.88	100	1.403	7.08	38.88	100	1.295	8.03	43.83	100
	LR+LL	1.506	2.05(69%)	31.88(12%)	18	1.457	2.19(66%)	33.81(6%)	18	1.457	2.19(69%)	33.81(13%)	18	1.433	2.37(67%)	33.72(13%)	20	1.288	5.44(32%)	44.43(-1%)	43
	ILP-1	1.524	**1.64(75%)**	**32.99(9%)**	8	1.492	6.5(0%)	38.91(-8%)	47	-	-	-	-	-	-	-	-	-	-	-	-
	ILP-2	1.516	2.79(57%)	34.46(5%)	18	1.493	**3.52(46%)**	**35.67(1%)**	21	1.459	**6.92(2%)**	40.21(-3%)	53	-	-	-	-	-	-	-	-
	Heur	1.536	2.48(62%)	33.77(6%)	17	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
viterbi	LR	0.755	1.63	157.37	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	LL	0.733	15.55	165.62	100	0.7	15.99	175.61	100	0.7	15.99	175.61	100	0.664	16.32	188.49	100	0.615	16.86	201.46	100
	LR+LL	0.74	4.71(70%)	157.86(5%)	10	0.71	7.03(56%)	170.08(3%)	21	0.643	7.03(56%)	187.43(-7%)	20	0.643	7.03(57%)	187.43(1%)	20	0.603	8.97(47%)	203.19(-1%)	29
	ILP-1	0.719	6.89(56%)	167.12(-1%)	25	0.719	6.89(57%)	**167.12(5%)**	25	0.687	7.51(53%)	170.2(3%)	36	0.666	10.2(38%)	**177.87(6%)**	53	0.611	**14.58(14%)**	202.07(0%)	60
	ILP-2	0.715	6(61%)	167.54(-1%)	18	0.715	6(62%)	167.54(5%)	18	0.703	6.15(62%)	**168.45(4%)**	21	0.664	**6.83(58%)**	179.53(5%)	23	0.629	15.23(10%)	200.76(0%)	54
	Heur	0.698	**5.53(64%)**	172.79(-4%)	15	0.698	**5.53(65%)**	172.79(2%)	15	0.698	**5.53(65%)**	172.79(2%)	15	-	-	-	-	-	-	-	-
M0-2	LR	1.144	0.08	5.69	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	LL	1.067	0.37	5.55	100	1.067	0.37	5.55	100	1.067	0.37	5.55	100	1.014	0.52	6.46	100	0.938	0.64	7.61	100
	LR+LL	1.057	0.33(11%)	5.69(-3%)	66	1.057	0.33(11%)	5.69(-3%)	66	1.057	0.33(11%)	5.69(-3%)	66	0.987	0.52(0%)	6.72(-4%)	72	0.897	0.57(11%)	7.75(-2%)	77
	Heur	1.073	**0.26(30%)**	6.42(-16%)	22	1.073	**0.26(30%)**	6.42(-16%)	22	1.054	**0.33(11%)**	6.62(-19%)	26	-	-	-	-	-	-	-	-
M3-2	LR	1.669	0.48	23.92	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	LL	1.555	1.05	21.84	100	1.555	1.05	21.84	100	1.555	1.05	21.84	100	1.477	1.31	23.39	100	1.384	1.63	25.81	100
	LR+LL	1.548	0.71(32%)	21.74(0%)	43	1.548	0.71(32%)	21.74(0%)	43	1.548	0.71(32%)	21.74(0%)	43	1.48	0.93(29%)	23.65(-1%)	49	1.389	1.41(13%)	26.42(-2%)	56
	Heur	1.625	**1.44(37%)**	25.7(-18%)	20	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

1) The overall outcomes, i.e., power and speed benefits from mixed-Vt, are strongly library- and design-dependent.
2) For Enable1 (generic library)-based designs, up to 20% speed improvement with 54% LL region is observed.
3) For Enable2 (rich library)-based designs, up to 7% speed improvement with 26% LL region is observed.

1) Designs Without Rich Library Cells (Enable1): For the designs implemented with *Enable1*, we observe that *M0* and *viterbi* achieve 20% speedup with ILP-based optimizations. For *M0*, ILP-1 achieves the *20% speedup* with 53% LL area. Compared to LR+LL (without placement constraints) in the ILP-1 solution, leakage, and total power values are 48% and 18% larger while the LL area is the same. For *viterbi*, ILP-1 and ILP-2 achieve the 20% speedup with 60% and 54% of LL area, respectively. Compared to LR+LL, in the ILP-2 solution, leakage power value is 69% larger, but total power value is 1% smaller, while LL area is 25% larger. For *ldpc*, ILP-2 achieves the 7% speedup target with 53% of LL area. Compared to LR+LL, in the ILP-2 solution, leakage and total power values are 101% and 20% larger, respectively, while LL area is 3% smaller.

We see that *ldpc* is not an SDP-friendly design, since the LL portion in LL+LR is higher compared to other designs. Even without placement constraints, ∼77% of area is needed to achieve the 20% speedup requirement. For *M3*, ILP-2 achieves the 7% speedup with 53% LL area. Compared to LR+LL, in the ILP-2 solution, leakage and total power values are 216% and 119% larger, while LL area is 35% larger. We believe that there are at least two reasons for not achieving $\geq$ 10% for *M3*. First, since *M3* has a relatively larger initial ECP compared to other testcases, it is more challenging to achieve a higher % of speedup. In particular, worst setup slack must be improved by more than 250 ps for the 20% speedup target, while other designs can meet such a speedup goal with less than 200 ps worst setup slack gain. Second, the placement seems to not be SDP-friendly. The required LL area values to achieve 7% speedup target without and with placement constraints are 18% and 53%, respectively, and there is a very large gap (i.e., 35% difference in LL area) between these two cases. This indicates that critical cells are indeed placed sparsely (that is, without any spatial contiguity awareness in physical synthesis or sizing steps) by commercial implementation flows, and that it is not easy to cover the critical cells with only a couple of rectangular LL regions.

In Fig. 4(a)–(h), we observe that Heur, ILP-1, and ILP-2 curves are between the LL and the LR+LL curves, but closer to the LR+LL curves. However, for faster ECPs (i.e., less than the minimum clock period achievable in pure-LR designs), the power values of the Heur, ILP-1, and ILP-2 increase dramatically. We note that the LR+LL results are obtained *without* considering placement constraints while the Heur, ILP-1, and ILP2 consider the placement constraints. The unnecessary cell swap to LL due to the placement constraints leads to large power increase.

2) Designs With Rich Library Cells (Enable2): For the designs implemented with Enable2, we observe that up to 7% and 2% speed improvement is achieved for *M0-2* and *M3-2*, respectively. In Fig. 4(i)–(l), we see that for designs with Enable2, as noted in Section V-A, the benefit of mixed Vt is relatively less. As expected, the Heur results do not show much power benefit (or a reduction) in comparisons are made at same ECP values (i.e., iso-ECP).

V. ON THE DIFFICULTY OF THE SDP PROBLEM

From our experimental results above, we see that the benefit of fine-grained mixed-Vt can be disappointingly small for

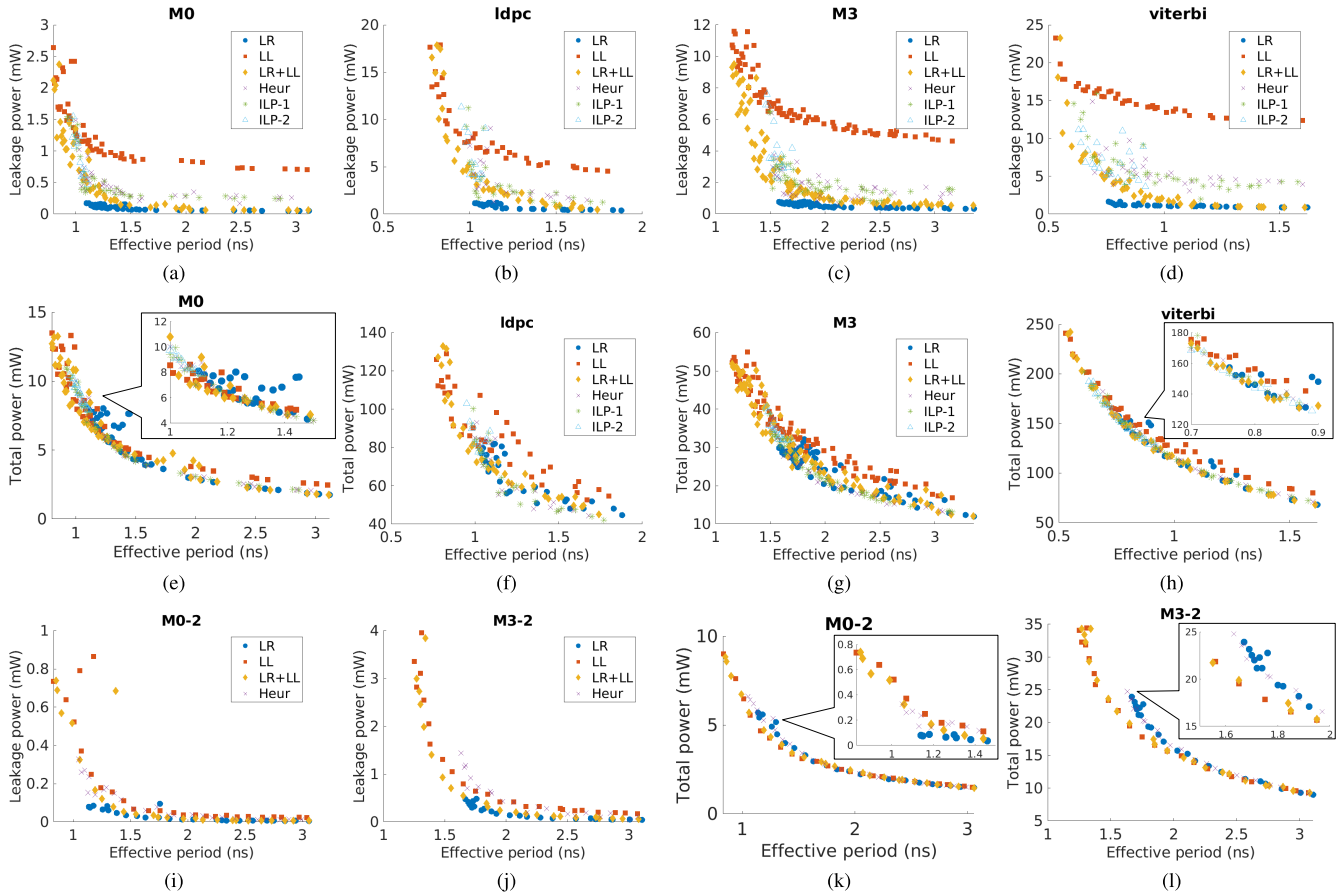


Fig. 4. (a)–(d) Leakage power versus effective periods for the four designs implemented with *Enable1*. (e)–(h) Total power versus effective periods for the four designs implemented with *Enable1*. (i) and (j) Leakage. (k) and (l) Total power versus effective periods for the two designs implemented with *Enable2*.

FDSOI implementations. In this section, we present what we believe to be root-cause, *intrinsic* reasons behind the difficulty of obtaining larger benefits from fine-grained mixed-Vt in FDSOI. These reasons stem from the nature of popular 28-nm FDSOI foundry technologies, as well as the input designs (netlist and placement), that we study. All of these reasons also apply to fine-grained body biasing in FDSOI.

In this section, our discussion is supported by experimental results obtained through limited access to another commercial enablement, which we refer to as *Enable3*. For Enable3, 22-nm 8T LL and LR that have six cell variants, respectively are used as library cells. Synopsys Design Compiler N-2017.09 [22] and Cadence Innovus v17.1 [17] are used for logic synthesis and P&R tools, respectively.

A. Rich Library Cell Options

We observe that the nature of foundry libraries can affect available mixed-Vt benefit. More specifically, if the foundry libraries offer rich cell library options that can be mixed without conflicts in the layout, the use of mixed Vt (which is restricted by placement constraint) would not give much benefit over single-Vt implementation.

Different poly bias options are an example. These options in the same-well-structure group that can be mixed in the layout without placement constraints, and they offer power-delay tradeoff wide enough to replace different well structure groups. The Enable2 28-nm FDSOI foundry enablement offers four poly bias options (P0, P4, P10, and P16) for each group of LL and LR. Similarly, the libraries in Enable3 offer six different cell options for each of LL and LR.

We have two experiments: 1) delay-power tradeoff of individual library cells and 2) delay-power tradeoff of design implementation to study the impact of the availability of rich cell library options on the benefit of mixed Vt.

1) Study of Individual Cell Delays: Fig. 5 shows leakage-versus-delay curves for different sizes of buffer cells for each Vt/poly bias option. The delay value of each cell is calculated using the lookup table in the nonlinear delay model (NLDM) library with input slew 50 ps and load of 4× the input capacitance of each cell. The average leakage power and the delay of each cell, respectively, correspond to the y-axis and the x-axis in the figure. We observe that the power-delay curves of LL and LR expand with the availability of more poly bias options, such that these curves have greater overlap and become more "near-continuous," with near-minimum power being achievable for a particular operating frequency *without* mixing LL and LR. Accordingly, the benefit of mixed-Vt is less likely to justify the overheads from placement constraints. (This might be in contrast to multi-Vdd or mixed-height placement contexts, where delay-power tradeoff curves remain disjoint and

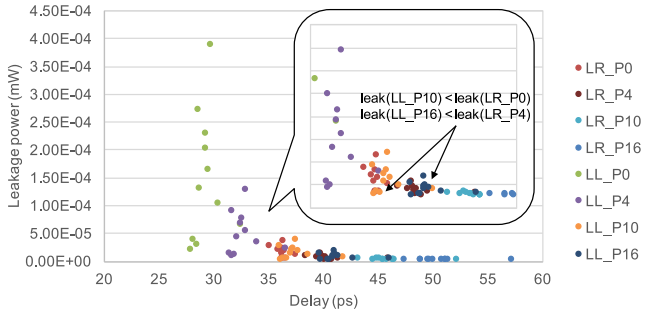


Fig. 5. Leakage versus delay curves of buffer cells with various Vt and poly bias options available in Enable2. The delay is measured with input slew 50 ps and output load of 4× the input capacitance of each cell. LL_P16 and LL_P10 consume less leakage power than LR_P4 and LR_P0.

mixing of flavors retains benefits.) Further, a reversed leakage trend is seen, i.e., leakage power of LL_P16 and LL_P10 is lower than that of LR_P4 and LR_P0 with iso-delay, in the zoomed-in region. With the availability of rich poly bias options, the benefit of fine-grained mixed-Vt might not be sufficient, let alone compelling, compared to "mixed poly bias" implementations.

B. Study of Design Implementations

Our experimental studies confirm that with rich cell library options, mixing of LL and LR might achieve only limited benefits. For example, Fig. 6 shows the power and delay tradeoff of different M3 implementations with different library cells, i.e., LL, LR, LR+LL (mixed-Vt) with Enable1, Enable2, and Enable3. Each dot corresponds to a distinct SP&R implementation with a target period. The *x*-axis shows the ECP in ns. Leakage or total power values are shown in the *y*-axis. We note that Enable1 is an enablement with generic cell library options, since only one cell option is available for each of LL and LR.

Notice that when a cell library option is limited, as in the case of Enable1, mixing Vt is beneficial especially for leakage power. However, with rich cell library options (Enable2 and Enable3), the benefit of mixing Vt is less. In Fig. 6, the red, yellow, and blue curves (dots) correspond to LL, mixed-Vt, and LR designs, respectively. For relatively slower ECPs (i.e., achievable by LR designs), LR always dominates in terms of leakage and total power. For relatively faster ECPs (i.e., less than the minimum achievable clock period), mixed-Vt dominates in terms of leakage and total power in Enable1. For Enable1 plots [Fig. 6(a) and (d)], the gap between the yellow and the red curves dots is clearly visible. However, such a trend is not observed with either Enable2 or Enable3. We also note that the plots in Fig. 6 do not consider placement constraints. The benefits of LL_LR designs may be obviated if placement constraints are considered. That is to say, when spatial contiguity constraints are considered in the placement, additional Vt swaps must be made to achieve legal placements, and hence the LL region will be larger than necessary.

C. Many Near-Critical Paths

Recall that mixed Vt in FDSOI requires "partitioning" the input placement/netlist in terms of speed since LL (fast) and

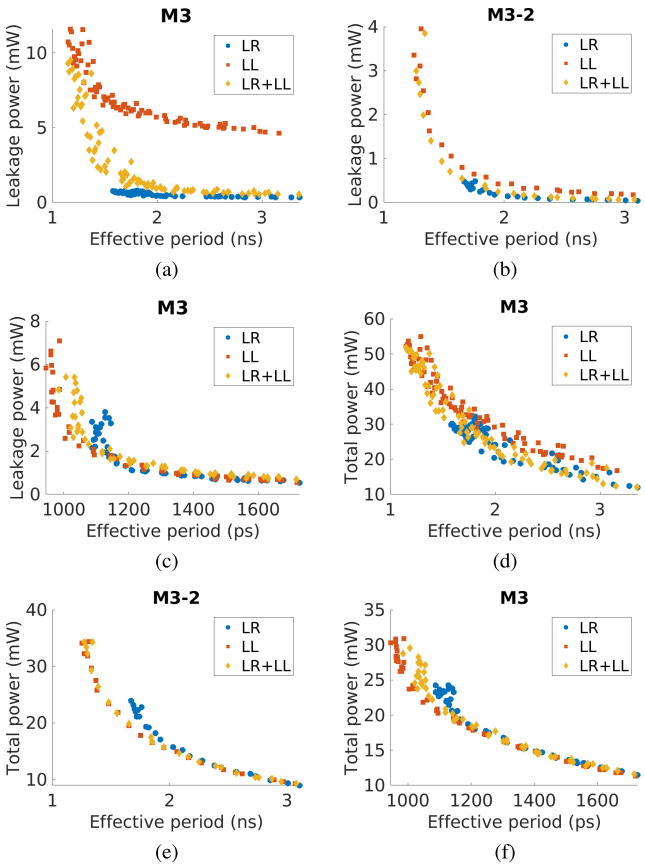
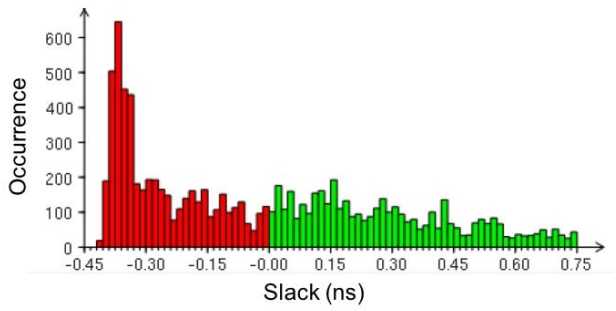


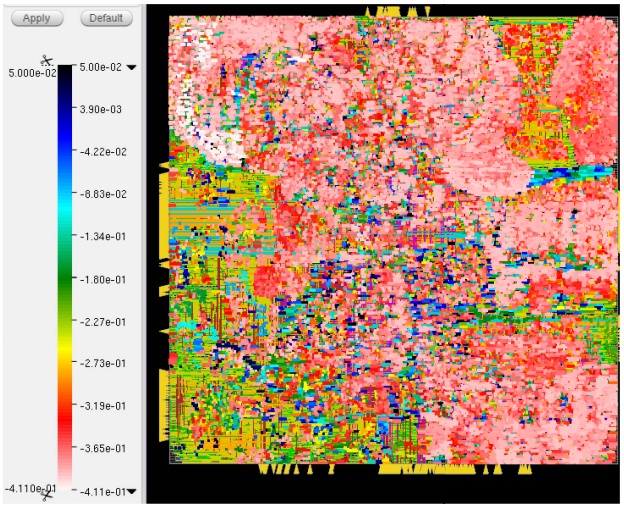
Fig. 6. Leakage power versus effective period curves for various M3 implementations with (a) Enable1, (b) Enable2, and (c) Enable3, along with total power versus effective period curves for various M3 implementations with (d) Enable1, (e) Enable2, and (f) Enable3. ECP is calculated as the target period subtracted by worst setup slack.

LR (slow) cells must be isolated. If a design has many timing-critical paths (which is quite common), SDP is inherently difficult to apply to the design. In many cases, designs with tight power and performance targets have many near-critical timing paths due to the optimizations performed during logic synthesis and placement. In such designs, a standard physical implementation flow will place timing-critical cells such that they are spread over the layout, which makes SDP more challenging.

Fig. 7(a) shows timing statistics of M3 implemented with Enable2. The target clock period of the M3 design is 2.1 ns, and timing is measured after post-placement optimization based on trial route. The worst setup slack is approximately −410 ps, which makes the ECP (i.e., the target clock period subtracted by the worst setup slack) 2.5 ns. In the figure, the *x*-axis and *y*-axis show the setup slack values and the occurrence of timing endpoints with the corresponding setup slack. We observe a typical slack wall, namely, that there is a high occurrence of timing endpoints near the worst (leftmost) slack value. Furthermore, due to the nature of optimizers that try to convert timing slack to minimize power, it is likely to see such a slack distribution (slack wall) after post-placement optimization. Designs with higher slack walls are more difficult to improve the worst slack, since more timing endpoints



(a)



(b)

Fig. 7. Timing information of M3 implemented with Enable2. (a) Histogram of path slack values, showing existence of wall of slack. 30% of paths must be fixed to achieve a 4% speed improvement. (b) Map of instance timing slacks of M3 implemented with Enable2, with legend shown in the left bar. White and red cells are timing-critical.

need to be improved for the clock period to change. For example, in Fig. 7, ∼30% of timing endpoints should be improved to obtain ∼100 ps slack improvement (which is only 4% of the ECP).

D. Placement Constraints

The flip-well structure in FDSOI is also a root cause of limited benefit in mixed-Vt implementation. To form rectilinear islands, it is inevitable to make unnecessary Vt swaps: if an LR cell instance must be swapped to LL, the neighbor cells of this target cell must be swapped as well. Fig. 8 shows example placements where region constraints restrict the benefit of mixed-Vt implementation for timing/power optimization. The small rectangles are timing-critical cells. The left cartoon shows the case where timing-critical cells are not placed in the LL region, and thus cannot be swapped to LL for speed improvement. The right cartoon shows the case where all timing-critical cells are placed in the LL region to achieve speed improvement, but at the cost of huge power increase.

Fig. 7(b) shows a timing slack map of M3 with *Enable2*. White and red cells can be considered as timing-critical, as seen in the left legend bar. We observe that the benefit of

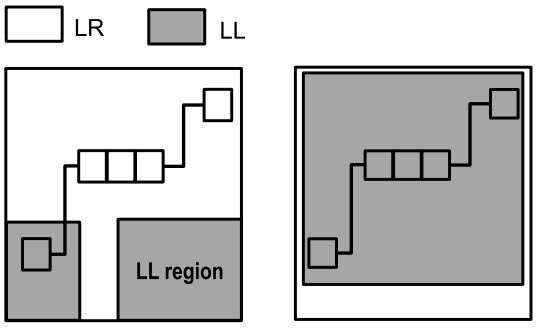


Fig. 8. Mixed-Vt with region constraints in a placement. The small rectangles are timing-critical cells. Left: Timing-critical cells are not covered by LL region, thus no speed benefit. Right: All timing-critical cells are covered by LL region, thus huge power increase due to the large LL region.

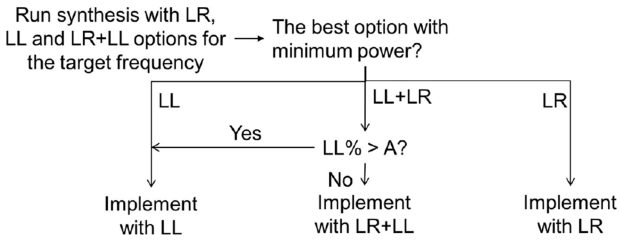


Fig. 9. Notional decision tree for FDSOI implementation option choice.

using LL, i.e., speed improvement, dramatically drops as we give more placement constraints. Based on our experiments, for the M3 design, 9% speed improvement is achievable by swapping 17% of the area to LL without considering placement constraints. However, with placement constraints, the speed improvement drops to 1% with a similar area of LL swaps (19%). We also have studied a variety of preplacement optimizations, but without success. More specifically, we collect all the timing-critical cells up front and place them locally, i.e., with region constraints. With the recent release of commercial P&R tools that we use, we find that this approach is too disruptive to conventional timing- and wirelength-driven placement, and that it leads to several suboptimal placement solutions with worse QoR in terms of both timing and wirelength.

VI. Conclusion

In this article, we have studied the potential of fine-grained mixed-Vt optimization in FDSOI. We formulate the SDP problem and propose effective heuristics that are capable of achieving significant speed improvements. We also identify inherent challenges that limit benefit from fine-grained mixed-Vt: 1) availability of rich cell library options in some commercial foundry enablements; 2) existence of a slack wall in well-optimized designs; and 3) spatial contiguity constraints (arising from well structure) in the placement. These challenges are confirmed in implementation experiments with multiple commercial enablements at 28 and 22 nm. Given our observations regarding sensitivity of mixed-Vt benefits to initial designs and library options, we offer a decision tree that may help designers make implementation choices, as follows.

A. Decision Tree

Fig. 9 shows our notional decision tree, based on our experimental studies and observations, for implementation option choice in FDSOI. For the input RTL, logic synthesis results with LL, LR and mixed-Vt are needed to see which implementation option offers the minimum power for the target operating frequency. If mixed-Vt is the best option, measure the portion of LL cells (LL%) in the synthesized netlist. If LL% $>$ A, it would be better to use LL option since we observe that mixed-Vt designs may not offer better power if LL cells are dominant. For A, we empirically recommend to use 15%. This is because LL% is likely to increase in the presence of placement constraints (in our experimental results, we observe a typical increase of $\sim 3\times$). Further, with $LL\% >\sim 50\%$, not much power benefit is seen compared to pure LL designs (i.e., LL% $= 100$).

B. Difficulty of Fine-Grained Body-Biasing in FDSOI

Last, we would like to add a brief further discussion regarding the potential for fine-grained *body-biasing* in FDSOI, and an additional fundamental challenge for this optimization. We first state the SDP problem for the body-biasing context, as follows.

1) Problem Formulation ("SDP-FBB") for Forward Body Bias: Given an initial placed design implemented with LR cells only, we perform Vt swapping, sizing and placement optimization to define "LLFBB" (i.e., LL with FBB applied) regions under timing/placement constraints. In this problem formulation, we would only consider FBB on LL since the feasible range of FBB voltage on LR is very limited.

Input: A placed design, synthesized, and optimized with LR cells.

Output: An optimized mixed-Vt netlist/placement, with FBB islands.

Constraints: The target operating frequency f_{fbb} at FBB mode should be $X\%$ higher than f_{nbb} [the operating frequency at zero/no body bias (NBB)]. The Δ total power $((p_{\text{fbb}} - p_{\text{nbb}})/p_{\text{nbb}})$ is no more than $Y\%$, where p_{fbb} (resp. p_{nbb}) is the total power at FBB (resp. NBB) mode. LL/FBB regions should be rectilinear islands, and the area should be less than $Z\%$ of the total area of the design.

The SDP-FBB problem has a fundamental, *moving baseline* challenge inherent in setting the baseline for speed boost target. This is because both the baseline f_{nbb} and the target f_{fbb} change during cell-swapping optimization (i.e., LLFBB island generation), as illustrated in Fig. 10. Thus, it is not straightforward to calculate a target frequency f_{fbb}. The moving baseline presents a chicken-egg situation: Once we generate an LLFBB island to improve timing by covering cells on the critical path, we can improve f_{fbb}. Meanwhile, f_{nbb} gets improved as well since cells in the LLFBB island automatically become LL. Eventually, an increased f_{nbb} sets a new target f_{fbb}, which induces a convergence issue.

C. Looking Forward

Finally, we believe that future work must further elucidate the cost-benefit tradeoffs in fine-grain, mixed-Vt (and, body biasing-based) FDSOI. This will be essential to correct technology adoption decisions by product teams. This article is only a first step toward this understanding. We believe that specific near-term research targets include identification of important parameters that determine SDP-friendly designs; consideration of clock distribution and on-chip variation in sign-off analyses; inclusion of useful skew into the optimization flows; hold time considerations; and improved optimization heuristics for the SDP implementation problem. Additionally, further direct investigations of the potential of fine-grained body biasing in FDSOI are required, e.g., by developing improved implementation flows with preplacement netlist and useful skew optimizations.

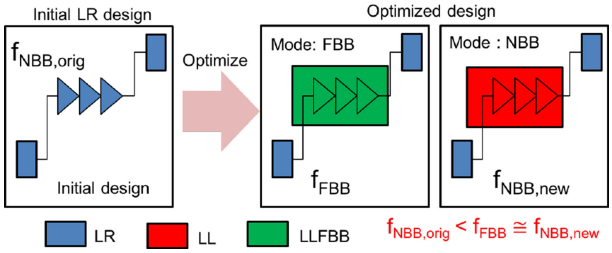


Fig. 10. "Moving baseline" challenge: As f_{fbb} is improved, the value of f_{nbb} changes during the process of LLFBB island generation.

ACKNOWLEDGMENT

The authors would like to thank Design and Research and Development Teams at NXP Semiconductors for their help in enabling the reported studies.

REFERENCES

[1] D. Albano, M. Lanuzza, R. Taco, and F. Crupi, "Gate-level body biasing for subthreshold logic circuits: Analytical modeling and design guidelines," *Int. J. Circuit Theory Appl.*, vol. 43, no. 11, pp. 1523–1540, 2015.

[2] A. C. Flores, "Layout floorplan using body bias islands," M.S. thesis, Dept. Elect. Eng., TU Eindhoven, Eindhoven, The Netherlands, Aug. 2010.

[3] S. Block *et al.* (2016). *Entering FD-SOI Era Using GLOBALFOUNDRIES 22FDX Technology*. [Online]. Available: https://www.globalfoundries.com/sites/default/files/articles/entering-fd-soi-era-using-globalfoundries-22fdx-technology.pdf

[4] P. Corsonello, M. Lanuzza, and S. Perri, "Gate-level body biasing technique for high speed sub-threshold CMOS logic gates," *Int. J. Circuit Theory Appl.*, vol. 42, no. 4, pp. 65–70, 2014.

[5] S. Dobre, A. B. Kahng, and J. Li, "Design implementation with noninteger multiple-height cells for improved design quality in advanced nodes," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 4, pp. 855–868, Apr. 2018.

[6] P. Flatresse. (2013). *UTBB-FDSOI Design & Migration Methodology*. [Online]. Available: http://cmp.imag.fr/IMG/pdf/utbb-fdsoidesign_migration_methodology_.pdf

[7] L. Guo, Y. Cai, Q. Zhou, and X. Hong, "Logic and layout aware voltage island generation for low power design," in *Proc. ASPDAC*, 2007, pp. 666–671.

[8] D. Jacquet *et al.*, "A 3 GHz dual core processor ARM Cortex TM -A9 in 28 nm UTBB FD-SOI CMOS with ultra-wide voltage range and energy efficiency optimization," *IEEE J. Solid-State Circuits*, vol. 49, no. 4, pp. 812–826, Apr. 2014.

[9] J. M. Kühn, H. Amano, W. Rosenstiel, and O. Bringmann, "Leveraging FDSOI through body bias domain partitioning and bias search," in *Proc. DAC*, 2016, pp. 1–6.

[10] B. Liu, Y. Cai, Q. Zhou, and X. Hong, "Power driven placement with layout aware supply voltage assignment for voltage island generation in dual-VDD designs," in *Proc. ASPDAC*, 2006, pp. 1–6.

[11] R. L. S. Ching, E. F. Y. Young, K. C. K. Leung, and C. Chu, "Post-placement voltage island generation," in *Proc. ICCAD*, 2006, pp. 641–646.
[12] R. Taco, I. Levi, M. Lanuzza, and A. Fish, "Low voltage logic circuits exploiting gate level dynamic body biasing in 28 nm UTBB FD-SOI," *Solid-State Electron.*, vol. 117, pp. 185–192, Mar. 2016.
[13] H. Wu, M. D. F. Wong, I.-M. Liu, and Y. Wang, "Placement-proximity-based voltage island grouping under performance requirement," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 7, pp. 1256–1269, Jul. 2007.
[14] H. Xiang *et al.*, "Row based dual-VDD island generation and placement," in *Proc. DAC*, 2014, pp. 1–6.
[15] H. Xiang *et al.*, "Gate movement for timing improvement on row based dual-VDD designs," in *Proc. ISQED*, 2016, pp. 423–429.
[16] C. Yeh, Y.-S. Kang, S.-J. Shieh, and J.-S. Wang, "Layout techniques supporting the use of dual supply voltages for cell-based designs," in *Proc. DAC*, 1999, pp. 62–67.
[17] *Cadence Innovus User's Manual*. Accessed: Aug. 22, 2019. [Online]. Available: http://www.cadence.com
[18] *Cadence Genus Synthesis Solution User's Manual*. Accessed: Aug. 22, 2019. [Online]. Available: http://www.cadence.com
[19] *Global Foundries: 22FDX*. Accessed: Aug. 22, 2019. [Online]. Available: https://www.globalfoundries.com/technology-solutions/cmos/fdx/22fdx
[20] *IBM ILOG CPLEX*. Accessed: Aug. 22, 2019. [Online]. Available: https://www.ibm.com/products/ilog-cplex-optimization-studio
[21] *OpenCores: Open Source IP-Cores*. Accessed: Aug. 22, 2019. [Online]. Available: http://www.opencores.org
[22] *Synopsys Design Compiler User Guide*. Accessed: Aug. 22, 2019. [Online]. Available: http://www.synopsys.com
[23] *TCL/TK Commands Manual*. Accessed: Aug. 22, 2019. [Online]. Available: http://www.tcl.tk/man/tcl8.4/TclCmd/contents.htm

Hamed Fatemi received the B.Sc. degree from the Electrical and Computer Engineering Department, University of Tehran, Tehran, Iran, in 1998, the M.Sc. degree from the K. N. Toosi University of Technology, Tehran, in 2001, and the Ph.D. degree in computer architecture from the Eindhoven University of Technology, Eindhoven, The Netherlands, in 2007.

He is an Innovation Leader/Department Manager with NXP Semiconductors, San Jose, CA, USA. He has authored and coauthored over 25 U.S. patents, scientific publications, and presentations. His current research interests include low-power design, multiprocessors, heterogeneous and reconfigurable systems, and variability tolerance design.

Andrew B. Kahng (M'03–SM'07–F'10) received the Ph.D. degree in computer science from the University of California at San Diego, San Diego, CA, USA, in 1989.

He is a Professor with the Computer Science Engineering Department and the Electrical and Computer Engineering Department, University of California at San Diego. His current research interests include IC physical design, the design-manufacturing interface, combinatorial optimization, and technology roadmapping.

Hyein Lee received the B.S. degree in electrical engineering from Yonsei University, Seoul, South Korea, the M.S. degree in electrical engineering from the Korea Advanced Institute of Science and Technology, Daejeon, South Korea, in 2009, and the Ph.D. degree in computer engineering from the University of California at San Diego, San Diego, CA, USA.

From 2009 to 2012, she was with the Design Technology Team, Samsung Electronics, Suwon, South Korea. She is a Research and Development Engineer with Synopsys Inc., Sunnyvale, CA, USA. Her current research interests include low-power design optimization and the design-manufacturing interface.

José Pineda de Gyvez (SM'04–F'09) received the Ph.D. degree from the Eindhoven University of Technology, Eindhoven, The Netherlands, in 1991.

He is a fellow with NXP Semiconductors, Eindhoven, where he coordinates Research and Development efforts on low power design technologies. His industrial responsibilities are positioned at the interface between design and technology. He also holds the professorship Resilient Nanoelectronics (part-time) with the Department of Electrical Engineering, Eindhoven University of Technology. This professorship fills a gap between industry and academia by bringing industrial knowledge into classrooms, and open innovation into NXP. He was a Faculty Member with the Department of Electrical Engineering, Texas A&M University, College Station, TX, USA. He has over 150 publications in the fields of low power IC design, analog signal processing, and design for manufacturability and test. He has coauthored 4 books, and has over 20 U.S. granted patents.

Dr. Pineda de Gyvez has been an Associate Editor for several IEEE TRANSACTIONS and is often involved in program and steering committees of international symposiums. He is also an Editorial Board Member of the JOURNAL OF LOW POWER ELECTRONICS.