Logic Design Partitioning for Stacked Power Domains

Kristof Blutman, Hamed Fatemi, Ajay Kapoor, Andrew B. Kahng, Jiajia Li, and José Pineda de Gyvez

***Abstract*— Energy and battery lifetime constraints are critical challenges to IC designs. Stacked power-domain implementation, which connects voltage domains in series, can effectively improve power delivery efficiency and thus improve battery lifetime. However, such an approach requires balanced currents between different domains across multiple operating scenarios. Furthermore, level shifter insertion, along with placement constraints imposed by power domain regions, can incur significant power and area penalties. To the best of our knowledge, no existing work performs subblock-level partitioning optimization for stacked-domain designs. In this paper, we present an optimization framework for stacked-domain designs. Based on an initial placement solution, we apply a flow-based partitioning that is aware of multiple operating scenarios, cell placement, and timing-critical paths to partition cells into two power domains with balanced cross-domain current and minimized number of inserted level shifters. We further propose heuristics to define regions for each power domain so as to minimize placement perturbation, as well as a dynamic programming-based method to minimize the area cost of power domain generation. In an updated floor plan, we perform matching-based optimization to insert level shifters with minimized wirelength penalty. Overall, our method achieves an excellent current balance across stacked domains with less than 10% discrepancy, which results in up to more than 2× battery lifetime improvements.**

***Index Terms*— Digital integrated circuits, low-power optimization, partitioning algorithms, physical design, power domains.**

I. Introduction

ENERGY and battery lifetime constraints induce new and critical challenges to IC designs, especially for mobile and Internet of Things applications. To achieve power autonomy in the era of a slowing Moore's law, new low-power techniques must be exploited. While many low-power techniques [9] have concentrated on the circuit side of system design, power management techniques have received growing attention due to the importance of power efficiency. Notably, the misalignment of battery voltages compared with scaled core voltages causes inefficiencies that present significant opportunities for power saving. In order to better align system on chip power domain voltages with battery voltages,

Manuscript received February 2, 2017; revised April 30, 2017; accepted June 9, 2017. Date of publication August 1, 2017; date of current version October 23, 2017. This paper was presented at the 2017 Asia and South Pacific Design Automation Conference. *(Corresponding author: Jiajia Li.)*

K. Blutman, H. Fatemi, A. Kapoor, and J. Pineda de Gyvez are with NXP Semiconductors, Eindhoven, 5656 AG, The Netherlands (e-mail: kristof.blutman@nxp.com; hamed.fatemi@nxp.com; ajay.kapoor@nxp.com; jose.pineda.de.gyvez@nxp.com).

A. B. Kahng and J. Li are with the University of California at San Diego, La Jolla, CA 92093 USA (e-mail: abk@ucsd.edu; jil150@ucsd.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2017.2729587

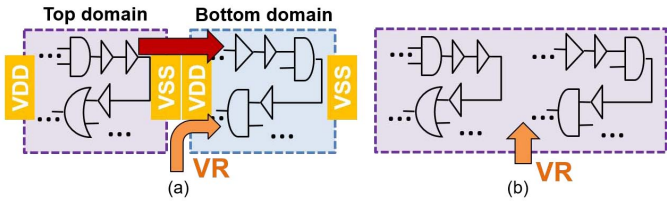


Fig. 1. Comparison between (a) stacked power-domain design versus (b) conventional design. VR indicates voltage regulator. The orange arrows indicate current from VRs. The red arrow indicates stacked current.

stacked power domain (or voltage stacking) has been proposed [21], [24], [27].

Fig. 1 shows the basic idea of stacked power domain. A stacked power-domain (or stacked-domain) design connects in series power domains that are connected in parallel in a conventional design.[1] Fig. 1 shows that one power domain (i.e., the top domain) is placed over the other (i.e., the bottom domain) to double the voltage and (ideally) halve the current compared with that in a conventional design. More specifically, if the supply voltage of a conventional design is V (i.e., $VDD = V$ and $VSS = 0$), then the $\{VDD, VSS\}$ of the top and bottom domains in the corresponding stacked-domain design are $\{2V, V\}$ and $\{V, 0\}$, respectively. For bulk CMOS, note that the $\{2V, V\}$ top domain must be placed on a deep n-well, so that the bulk potentials can be maintained at $(2V, V)$. Moreover, in the ideal case, the current is balanced across the two domains. The stacked-domain scheme provides implicit 2:1 downconversion of external supplies. In light of this, there is no need to employ a bulky supply to generate the supply voltage for the core (i.e., gate instances and memories). Instead, it suffices to employ a much smaller converter that acts only as a watchdog to the supply rail that connects the power domains. This results in increased power efficiency for the overall system [4].

Based on the power conversion modeling proposed in [3], we derive the battery lifetime improvement from stacked-domain optimization as follows. (Table I lists the notations used in our discussion.) Since battery lifetime (T) is inversely proportional to P_{ext}, we compare P_{ext} of a

[1]This paper focuses on optimization with two power domains (i.e., top and bottom domains) and conventional planar monolithic implementation (as opposed to 3-D integration). In other words, both stacked-domain and conventional designs in the following discussions are planar monolithic implementations. Stacked-domain optimization with more than two power domains and/or in 3-D integrated circuits is left as a direction for future research.

TABLE I
DESCRIPTION OF NOTATIONS USED IN OUR DISCUSSION

Term	Meaning
P_{ext}	total input power from external supply (e.g., battery)
$P_{VR,in}$	input power of voltage regulator from external supply
$P_{VR,out}$	output power of voltage regulator to core
η_{VR}	power conversion efficiency of voltage regulator
P_{stk}	direct power of stacked power domain from external supply
P_{core}	total power consumption of core
I_{stk}	stacked current (current from top domain to bottom domain)
I_{VR}	output current from voltage regulator
T	battery lifetime

stacked-domain design to that of a conventional design.[2] By definition (see Table I), in a stacked-domain design, we have

$$P_{\text{ext}} = P_{\text{stk}} + P_{VR,in} = P_{\text{stk}} + P_{\text{VR,out}}/\eta_{\text{VR}}. \tag{1}$$

On the other hand, in a conventional design, power is only supplied through the VR. Thus, the total input power from the external supply of a conventional design (P'_{ext}) is calculated as

$$P'_{\text{ext}} = P_{\text{core}}/\eta_{\text{VR}}. \tag{2}$$

By assuming the same core power consumption in both stacked-domain and conventional designs, we have

$$P_{\text{core}} = P_{\text{stk}} + P_{\text{VR,out}}. \tag{3}$$

Furthermore, based on the model described in [3], which assumes that the VR power efficiency is the same for both cases, we have

$$P_{\text{VR,out}}/P_{\text{core}} = I_{\text{VR}}/(2 \cdot I_{\text{stk}} + I_{\text{VR}}). \tag{4}$$

Finally, based on the above-mentioned analyses, the battery lifetime ratio between the stacked-domain design (T) versus the conventional design (T') is [3]

$$\frac{T}{T'} = \frac{P'_{\text{ext}}}{P_{\text{ext}}} = \frac{P_{\text{core}}/\eta_{\text{VR}}}{P_{\text{stk}} + P_{\text{VR,out}}/\eta_{\text{VR}}} = \frac{(2 \cdot I_{\text{stk}} + I_{\text{VR}})/\eta_{\text{VR}}}{2 \cdot I_{\text{stk}} + I_{\text{VR}}/\eta_{\text{VR}}} = \frac{2 \cdot I_{\text{stk}} + I_{\text{VR}}}{2 \cdot \eta_{\text{VR}} \cdot I_{\text{stk}} + I_{\text{VR}}}. \tag{5}$$

We observe that the battery lifetime benefit from a stacked-domain implementation increases with a smaller I_{VR}. As a motivating example, if the current is perfectly balanced between two domains (i.e., $I_{\text{VR}} = 0$), assuming $\eta_{\text{VR}} = 80\%$, the stacked-domain implementation provides 25% battery lifetime improvement over the conventional implementation. Moreover, since the power efficiency of the VR decreases with small supply currents, the battery lifetime benefit from the stacked-domain implementation is expected to be higher for designs in low-power modes that use a VR optimized for high-power cases.

Although the stacked-domain implementation provides significant battery lifetime improvement, it also raises nontrivial implementation methodology challenges that must be solved. First, the communication between the power domains must be ensured by level shifters that can convert such extreme signal levels. Second, the power efficiency improvement is directly dependent on the current balancing between the two power domains. In other words, the design must be bipartitioned in terms of current consumption. We also note that such a partitioning optimization must comprehend multiple operating scenarios, area, and power penalties as well as timing impact of level shifters, as well as additional placement constraints imposed by region definition of power domains. The first challenge has been thoroughly investigated, with several different level shifter architectures having been proposed [24], [27]. However, the optimization of partitioning and layout planning of the designs has remained an open challenge that prior works (which have mostly been ad hoc or design-specific) do not ultimately answer for general systems. In this paper, we address this open challenge and provide a comprehensive optimization framework for partitioning and floor planning of stacked-domain implementation that can be used for a wide range of systems.

The contributions of this paper are as follows.

1) We propose a comprehensive optimization framework for stacked-domain implementation. Key elements include a flow-based partitioning with layout and timing-path awareness, heuristics for layout region generation of power domains, and a matching-based optimization for level shifter insertion.
2) We are the first to propose a partitioning optimization at the subblock level for stacked-domain implementation that can be used for a wide range of systems.
3) We validate our optimization flow on industrial designs, in the context of an industrial implementation flow that includes placement, clock tree synthesis, and routing.
4) Our optimization accommodates current balancing constraints between stacked domains, and multiple partitioning scenarios with respect to movement of hard macros and logic across the stacked domains.
5) Using the power delivery block described in [4], our optimized stacked-power domain designs achieve more than 10% and 2× battery lifetime improvement compared with the conventional designs in function and sleep modes, respectively.

II. PREVIOUS WORK

In this section, we review the previous literature on: 1) stacked-domain implementation and 2) netlist partitioning. Our stacked-domain optimization problem is different from the power-island generation problem [8], [13], [28], in that the power-island generation optimization assumes different supply voltages for power domains and minimizes the power overhead from voltage assignments, while our optimization exploits charge recycling by balancing current across domains. Moreover, many critical issues, such as timing impact of level shifters, insertion of shifter rows, and region definition of power domains, are not addressed in power island-related works.

A. Stacked-Domain Implementation

The circuit blocks needed for a stacked-domain implementation—such as level shifters and VRs—are well

[2]We use battery lifetime (T) as the metric to evaluate energy improvement achieved by our proposed methodology. We also report power values of core (P_{core}) and the entire system (P_{ext}) from our optimization in Table III.

studied in the literature. However, to the best of our knowledge, no existing work is able to fully automate the implementation flow of a stacked-domain design. Various VRs and level shifters have been studied in [24] and [27], but the designs used in their studies lack the complexity of a realistic application. A smart regulation scheme has been proposed in [21], and the studied design has relatively higher complexity, featuring processor cores. At the same time, in the work of [21], there is no connection between different processor cores, which makes the partitioning problem much simpler. Similarly, [5] and [22] only focus on specific design Blocks, such as IO cells and memories. A recent work [3] applies stacked-domain optimization to a complete microcontroller unit (MCU) system designed with a standard design flow. The partitioning approach presented in [3] is somewhat ad hoc, and is not applicable to a general design. By contrast, here, we present a comprehensive optimization framework for stacked-domain implementation that is applicable to a wider range of designs.

B. Netlist Partitioning

As a classic problem in VLSI optimization, netlist partitioning has been thoroughly studied in the previous literature. A comprehensive, still-relevant taxonomy of approaches is given in [1]. We highlight four basic partitioning approaches.

1) Move-Based Approach: Fiduccia and Mattheyses [11] and Kernighan and Lin [19] propose to iteratively move or swap vertices guided by gain functions to partition a given set of vertices into two partitions with balanced weights and minimized number of hyperedge cuts. Important improvements and/or extensions have been proposed, such as multilevel extension [6], timing-path awareness [16], multiway partitioning [17], and "lookahead" gain functions (e.g., gain vectors, cluster-oriented iterative improvement partitioner/cluster-detecting iterative improvement partitioner, and last-in-first-out gain buckets [10], [14], [20]).

2) Mathematical Programming-Based Approach: Shih and Kuh [26] formulate the partitioning problem as quadratic Boolean programming to minimize the total cost of cell-to-partition assignments as well as the number of cuts, with respect to capacity and timing constraints. Goemans and Williamson [12] use semidefinite programming for partitioning optimization.

3) Flow-Based Approach: Yang and Wong [29] propose to use repeated max-flow computations and clustering operations to achieve a balanced bipartitioning solution. The work of [7] documents high efficiency and relatively good solution quality of flow-based partitioning with a min-cut objective.

4) Clustering Approach: Rajaraman and Wong [25] propose a clustering approach to minimize the delay from primary input (PIs) to primary output (POs) with a maximum-area constraint for each cluster.

In this paper, we apply the flow-based approach [29] to partition instances into two power domains. We propose several extensions to the existing flow-based partitioning, including layout and timing-path awareness, multiscenario weight (i.e., current) balancing, and a prior clustering step for runtime reduction.

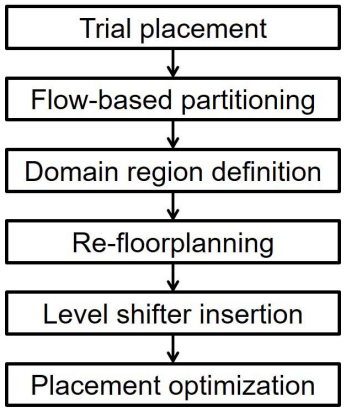


Fig. 2. Overall optimization flow.

III. METHODOLOGY

We now describe our optimization framework for stacked-domain logic design partitioning implementation. We first state our stacked-domain optimization problem as follows.

Given: A netlist, timing constraints, level shifters, VR efficiency, and switching information of instances in the netlist.

Do: Partition the netlist instances into two domains, define the layout region of each domain, and place instances and level shifters, such that battery lifetime is maximized.

As implied by (5), to maximize the battery lifetime, our basic objective is to balance the current between the two stacked power domains, while minimizing the power penalty due to level shifter insertion.

Fig. 2 shows our overall optimization flow. A common practice in stacked-domain implementations is to partition the netlist (i.e., define the power domain of each instance or block) prior to the floor-planning stage [3]. However, performing a power domain assignment before placement can result in sub-optimal floor-plan and placement solutions. More importantly, the placement optimization inserts buffers and upsized cells, which can change the current profile of each power domain. As a result, currents that have been balanced during the partitioning stage are no longer balanced after the placement stage. To resolve this, we propose to perform a *trial placement*, based on which we perform a layout-aware partitioning (with minimized number of cuts as well as placement perturbations) to assign instances to power domains.

Fig. 3(a) shows an example of our layout-aware partitioning solution on design advanced encryption standard (AES) [32] in 28LP technology. Based on the partitioning solution, we define the layout region for each power domain such that each domain has a continuous region [Fig. 3(b)]. Note that since gaps must be inserted along the boundary between two power domains, we propose a dynamic programming optimization to minimize the boundary length between two domains. We then legalize instance placements within the (updated) region for each power domain using a commercial place-and-route (P&R) tool [31]. We then update the floor plan by shifting the power domains

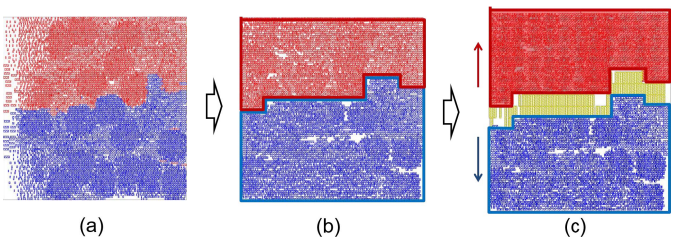


Fig. 3. Example of optimization. (a) Layout-aware partitioning; (b) region definition of power domains; and (c) level shifter insertion in the updated floor plan. Blue: instances assigned to the bottom domain. Red: instances assigned to the top domain. Yellow: level shifters. Design: AES (∼11K instances). Technology: 28LP.

[as shown in Fig. 3(c)] and inserting level shifters.[3] We perform a matching-based optimization to determine level shifter placement locations that minimize wirelength. Although our partitioning flow is aware of the timing-critical paths from the trial (initial) placement, level shifter insertions and placement legalization (according to defined power domains) can result in timing violations. We, therefore, perform an incremental placement optimization (including gate sizing and VT-swapping) to fix timing violations (last step in Fig. 2).

A. Flow-Based Netlist Partitioning

The greedy iterative partitioning approach is not naturally amenable to timing path-aware partitioning, and has no mechanism to preserve solution structure of an initial (trial) placement. We thus apply the flow-based approach described in [29] to partition instances into two power domains. Our objectives include: 1) to minimize the number of cuts, which reduces timing, area, and power penalties from level shifter insertions and 2) to minimize the perturbation to the initial placement solution.

To construct the flow network, following the approach in [29], we insert a vertex (V_i^c) for each cell or cluster of cells. For each net, we insert two vertices (V_j^n) and a bridging edge of unit capacity between the two vertices. For each cell or cluster of cells incident to a net, we insert two edges of infinite capacity between the vertex corresponding to the cell or cluster (V_i^c) and each of the two vertices corresponding to the net (V_j^n). Finally, the weight of a vertex corresponding to a cell or a cluster (V_i^c) is estimated based on the current of the cell or cluster, while the weights of vertices corresponding to nets (V_j^n) are set as zero. Fig. 4 shows an example of the constructed flow network corresponding to a net connecting two cells. More details of the flow network construction are described in [29]. As simplified illustrations, in the following discussions, we only show graphs of logic connections [see Fig. 4(a)] to describe our flow-based partitioning.

According to the max-flow min-cut theorem, the approach finds the partitioning solution with the minimum number of cuts for a given netlist via a max-flow optimization. However, this does not guarantee that the balancing constraint is met. To address this, after each max-flow optimization, the approach clusters the vertices belonging to the smaller partition together with one neighbor vertex (to avoid obtaining the same partitioning solution) into one super vertex. Based on the updated flow network, another max-flow optimization is performed. The approach iteratively performs (incremental) max-flow optimization and clustering until the balancing constraint is satisfied. In other words, the iterative max-flow optimization and clustering procedure keeps track of the aggregated current until the currents of two partitions are balanced. Fig. 5 shows the basic idea of the flow-based partitioning.

[3]We understand that modification of the block size might not be consistent with certain implementation flows. At the same time, we believe that performing the initial trial placement (with appropriate instance bloating) in a block having the shape of Fig. 3(c) will not diverge significantly from the initial trial placement in Fig. 3(a), particularly with improved (smaller) level shifter designs. Ongoing work is aimed at a predictive (or, "one loop") methodology to determine the block size prior to trial placement.

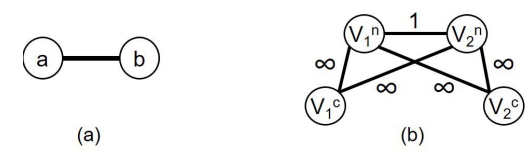


Fig. 4. Example of flow network construction for a net with degree of two. (a) shows a net connected with two cells a and b. (b) shows the corresponding flow network, where V_1^n and V_2^n are vertices corresponding to the net; V_1^c and V_2^c are vertices corresponding to cells a and b, respectively. Labels on the edges are capacities.

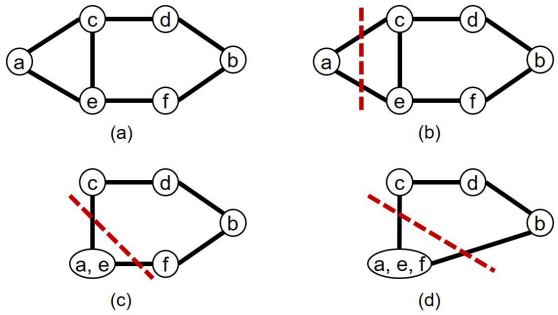


Fig. 5. Flow-based partitioning. (a) and (b) Source and sink, respectively. All vertices have the same weight. Red dotted lines: cuts. (a) Initial flow network. (b) First max-flow min-cut computation. (c) Clustering operation. (d) Second max-flow min-cut computation.

Algorithm 1 Flow-Based Partitioning

```
Pre-cluster cells into clusters
Define source and sink vertices
ΔI ← +∞
while ΔI > Δ_max do
    Perform max-flow optimization to achieve the min-cut solution
    Remove outliers (layout awareness)
    Remove V-shaped vertices (timing-path awareness)
    Cluster the smaller partition and one neighbor
    Update ΔI
end while
```

We adopt the flow-based partitioning approach to our stacked-domain optimization with the following five extensions. Algorithm 1 describes our partitioning procedure.

1) Extension 1 (Source and Sink Selection): The approach in [29] randomly picks two nodes (instances) in the flow network (netlist) as the source and sink nodes. However, there are cases in which the flows between selected source and sink vertices cannot cover the entire flow network, resulting in unbalanced partitioning solutions. As an example, the selection of vertices *a* and *b* as the source and sink from the flow network shown in Fig. 6(a) will not be able to achieve a balanced partitioning solution. To address this, we add a supersource

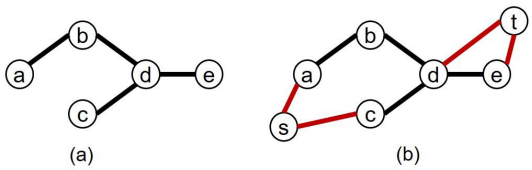


Fig. 6. (a) Choosing a/b, or c/d, or d/c as source/sink cannot lead to a balanced solution. (b) Adding a supersource (s) and a supersink (t) resolves the issue. Edges in black have unit capacities. Edges in red have infinite capacities.

and a supersink and connect them to multiple vertices (e.g., PIs and POs in a netlist, or instances located at the core boundary) with edges of infinite capacity to minimize the number of uncovered vertices (instances) as shown in Fig. 6(b).

2) Extension 2 (Layout Awareness): To avoid excessive placement perturbation, which can result in current profile changes and thus power penalty, the partitioning optimization must be aware of trial placement locations of the instances—such that instances partitioned into the same power domain are placed close to each other in the original trial placement. We achieve this required layout awareness in two ways. First, we only select the instances located close to each other to connect to the supersource (or supersink). As an example, we select instances located within a particular distance (e.g., ten cell rows) from the bottom (resp. top) core boundary to connect to the supersource (resp. supersink). Second, after each max-flow optimization, we detect *outliers*, which are instances belonging to the larger-current partition that are located within a region with a majority of instances belonging to the smaller-current partition. We then cluster these outliers with the instances from the smaller-current partition (line 6 in Algorithm 1).

3) Extension 3 (Critical-Path Awareness): Ignoring signal flow direction and timing path structure during the partitioning optimization can easily result in multiple cuts along one timing path. We extend the partitioning flow in [29] to minimize the number of cuts along timing-critical paths. Similar to the layout awareness extension discussed earlier, after each max-flow optimization, we detect "V-shaped vertices" [16], which are a sequence of instances belonging to the larger-current partition along a timing-critical path, where the fan-in and fan-out instances of these instances along the timing-critical path are in the smaller-current partition. We then collapse (cluster) the instances corresponding to the V-shaped vertices into the smaller-current partition as long as this does not violate the balancing constraints (line 7 in Algorithm 1).

4) Extension 4 (Preclustering): Although the max-flow optimization can be achieved with an incremental flow computation and the entire optimization takes $O(N)$ iterations to converge, where N is the number of instances in a design, the runtime in practice can be substantial for a large design. To reduce the runtime, we perform a preclustering optimization based on the heavy-edge matching (HEM) strategy [18]. We enforce layout-awareness constraints (i.e., an upper bound on the distance between two vertices that can be clustered) during the HEM. Fig. 7 shows an example of the HEM clustering up through 18 levels (clustering ratio = 0.76 at each level), showing how instances within the same cluster

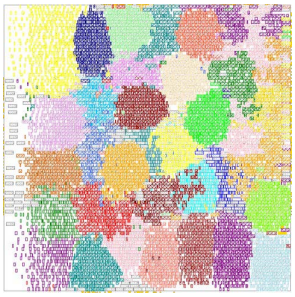


Fig. 7. HEM clustering solution. Different clusters are indicated by different colors. (But since we are limited by 64 available colors, different clusters can have the same color. Also, clusters with small size might not be visible from the figure.) #Clusters = 200. Levels of clustering = 18. Clustering ratio at each level = 0.76. Design: AES. Technology: 28LP.

are spatially proximate. Our experimental results show that we can reduce runtime by 75% (two HEM levels and overall clustering ratio of 0.5) with negligible degradation of solution quality (e.g., cut number).

5) Extension 5 (Multiple Operating Scenarios): To ensure high power efficiency across different operating scenarios, the partitioning optimization must balance currents between two domains across multiple scenarios (e.g., function mode with different input vectors, and sleep mode). To achieve this, we use the weighted sum of normalized currents from different scenarios during our optimization (line 9 in Algorithm 1). Specifically, the delta current is calculated as

$$\Delta I = \sum_i \left(w^i \cdot \left| I^i_{\text{top}} - I^i_{\text{bot}} \right| \; / \; \left(I^i_{\text{top}} + I^i_{\text{bot}} \right) \right) \tag{6}$$

where I^i_{top} and I^i_{bot} are, respectively, the currents of top and bottom domains in the ith mode, and w^i is the weighting factor of the ith mode, such that $\sum_i w^i = 1$.[4] Our optimization ensures that ΔI does not exceed a predefined upper bound (i.e., $\Delta_{\max}$ in Algorithm 1).

B. Domain Region Definition

In this section, we describe our methodologies to define the layout region (power island) for each power domain. The definition of the layout region for each power domain affects the design quality in two fundamental ways. First, gap area must be inserted along the boundary between different power domains. Therefore, a longer boundary length will lead to higher area penalty. Second, the power domain definitions will have downstream impact on the power delivery network (PDN) design, which is not yet implemented at this point. Therefore, it is desirable to adjust the power domain definitions for minimized area, power, and performance penalties.[5] If the partitioning and the trial placement results in discontinuous power domains, the length of the power domain boundaries is highly likely to be longer compared with the case when the regions of each power domain are merged. Moreover, the power routing will be more difficult, since different power rails will need to be routed to discontinuous power domain

[4] A *mode* is an operating scenario, such as sleep mode or function mode.

[5] Since the power domain definitions change after our partitioning optimization, in our implementation flow, we perform refloor planning with updated power grids.

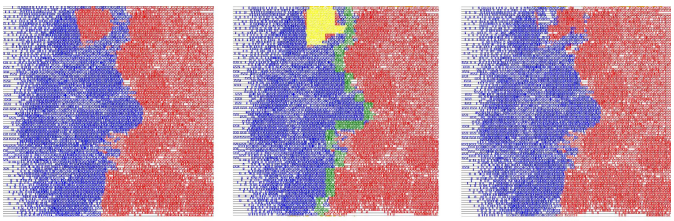


Fig. 8. FM-based bin movement. (a) Initial placement solution. Red and blue: instances partitioned to top and bottom domains. (b) Yellow: outliers of the top domain. Green: neighboring bins of the top domain. (c) Postmovement placement, where each domain has a continuous region. (The small number of remaining outliers is minority instances in their bins, and will be legalized during an incremental placement.) Design: AES. Technology: 28LP.

Algorithm 2 FM-Based Bin Movement

1: $U \leftarrow$ find outliers; $H \leftarrow$ find neighboring bins
2: **for all** $u \in U$, $h \in H$ **do**
3: **if** $u.domain \neq h.domain$ **then**
4: $cost(u, h) \leftarrow$ half-perimeter wirelength increase by swapping u and h
5: **else**
6: $cost(u, h) \leftarrow +\infty$
7: **end if**
8: **end for**
9: **while** $U \neq \emptyset$ **do**
10: $(u', h') \leftarrow Min_{cost(u,h)}\{(u, h) \mid u \in U,\ h \in H\}$
11: swap u' and h'
12: **if** create new outliers **then**
13: revert the swap; $cost(u', h') \leftarrow +\infty$
14: **end if**
15: update U, H and costs
16: **end while**

regions. Thus, we seek to have only two regions (i.e., power islands) corresponding to the two power domains.

Although our partitioning optimization is layout-aware, there can still be separated regions for each power domain. Fig. 8(a) shows an example trial placement and partitioning solution where the top domain (shown in red) has two separated regions. To merge the regions while minimizing placement perturbation (e.g., wirelength increase), we perform an FM-based bin movement optimization (i.e., an iterative, swap-based greedy algorithm as described in Algorithm 2). We first divide the core area into bins. The power domain of each bin is defined as the power domain of majority instances within the bin. We then define the outliers (i.e., bins outside the largest continuous region of the corresponding domain) and neighboring bins (i.e., bins adjacent to the largest continuous region of the different domain) (line 1). Fig. 8(b) shows an example of outliers and neighboring bins. We calculate the cost to swap pairs of outliers and neighboring bins (lines 2–8). We iteratively swap the pair of an outlier and a neighboring bin with the minimum movement cost, until all outliers [e.g., yellow bins in Fig. 8(b)] are removed (lines 9–16).

In the last step of domain region definition, we apply dynamic programming to minimize the length of the boundary between two power domains while maintaining the area within each domain. As the base cases, we calculate the boundary length decrease of each boundary segment by simplifying the boundary shape (e.g., highlighted segment in Fig. 9). We note that such simplification must meet an upper bound of moved area (i.e., total area with changed domain assignment). Assuming that the (turning) points along the boundary are

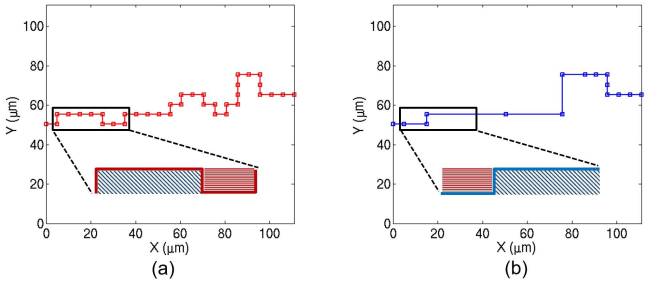


Fig. 9. Boundary optimization. (a) Original boundary between two power domains after bin movement. (b) Optimized boundary with smaller length. An example of segment optimization is shown. Optimized segments have smaller total length while maintaining the same area in each power domain. Design: AES. Technology: 28LP.

indexed from left to right or from bottom to top, the recurrence relation in our dynamic programming optimization is

$$\text{Sol}(j) = \text{Min}(\text{Sol}(i).\text{length} + \text{seg}(i, j).\text{length}), \quad \forall 1 < i < j \tag{7}$$

where $\text{Sol}(j)$ is the optimized boundary solution from the first point to jth point, and $\text{seg}(i, j)$ is the simplified boundary segment between the ith and the jth points. The dynamic programming-based boundary simplification has $O(M^2)$ time complexity, where M is the number of points or segments. The time complexity further decreases to $O(M)$ if we only search a limited range of existing subsolutions [i.e., i in (7)].

C. Level Shifter Insertion

In the last step of our optimization, we insert and place required level shifters between the top and bottom domains, and perform refloor planning if the total cell area exceeds the block floor-plan area. Specifically, we define placement regions for level shifters, where each region must have an even number of level shifter rows due to deep n-well sharing. Furthermore, we assume that the layout of the level shifter has already included the boundary of the deep n-well of either or both power domains, and that the edges facing either of the power domains have a standard-cell row structure. As a result, we are able to seamlessly integrate the level shifters with only a small separation (i.e., 2.5 μm) at left and right ends of each level shifter row from standard cells. The row height of our level shifter is 6× of the standard-cell row height.[6] Furthermore, additional space (i.e., ∼10% of area within each level shifter placement region) is required for tie and decap cell insertion. The objectives for our level shifter placement optimization are to minimize the area overhead and minimize the wirelength penalty due to level shifter region definition and level shifter placement, respectively.

Since the level shifter insertion approach in [2] does not comprehend the above-mentioned layout constraints as well as the space for tie/decap cell insertion, it cannot be applied in a realistic implementation of a stacked-domain design. We, therefore, propose a new level shifter placement approach. Algorithm 3 shows our level shifter placement procedure. We first perform a matching optimization (using the Hungarian algorithm [30]) to map each inserted level shifter to

[6]Our level shifter model is from our industry collaborators.

Algorithm 3 Level Shifter Placement

1: Matching optimization between candidate locations and level shifters
2: Level shifter placement
3: Clustering of level shifters to define level shifter placement regions
4: Placement blockage insertion and candidate locations update
5: Matching optimization between updated candidate locations and level shifters
6: Level shifter placement
7: Clumping level shifters to create space for tie / decap cell insertion
8: Legalization of standard cells

a candidate placement location (i.e., a level shifter placement site near the boundaries of power domains) with minimized wirelength (line 1). More specifically, we enumerate possible placement locations [i.e., all valid placement sites within the minimum possible (feasible) even number of rows] near the boundaries between two domains and calculate the potential cost [i.e., half-perimeter wirelength (HPWL) increase] of placing each level shifter onto each candidate placement location. Since cuts (i.e., level shifter insertions) occur on nets connecting top and bottom domains, placing level shifters along the boundaries between two domains will be less likely to cause routing detour. We define the cost as the total HPWL of nets connected to the level shifter. Based on the cost matrix, we perform matching optimization to assign the placement location for each level shifter while minimizing the total cost. Note that such a level shifter placement solution does not honor the layout constraint where each region must contain an even number of level shifter rows. We, therefore, cluster level shifters that are separated by a distance that is smaller than a predefined value (e.g., 20 μm) and create a region having an even number of rows for each cluster of level shifters (line 3).

According to the clustering solution, we generate placement blockages for standard cells and update candidate locations for level shifter placement (line 4). Specifically, we first round the height of the bounding box of the corresponding level shifter cluster to be the nearest even multiple of the level shifter row height, i.e., we satisfy the constraint of having an even number of level shifter rows. We use the rounded value as the height of the blockage. We then calculate the width of the blockage based on the total area of level shifters, the required spacing area, and the blockage height. Last, we shift the blockage horizontally to be centered at the weighted center, i.e., centroid of the level shifter cluster. Importantly, we also comprehend spacing requirement at the ends of each level shifter row during placement blockage insertion. We then perform another iteration of matching optimization based on the updated candidate placement locations, and place level shifters accordingly (lines 5 and 6). Observe that in the above-mentioned optimizations, we use level shifters with bloated (e.g., by 10%) widths. We now recover the level shifters' cell widths and clump them to create space for tie/decap cell insertion (line 7). Finally, we perform placement legalization of standard cells to move them out from the created level shifter placement regions. Fig. 10 shows an example of level shifter placement.

IV. Experimental Results

We perform experiments in a 28-nm LP foundry technology with dual-VT libraries. We use four design blocks (AES, data encryption standard (DES), JPEG, and video graphics array (VGA)) from OpenCores [32] as our test cases. Parameters of these four test cases are shown in Table II.[7] The worst case timing and power analysis view for AES, DES, JPEG, and VGA is (SS, 0.95 V, 125 °C). We synthesize designs using Synopsys Design Compiler vI-2013.12-SP3 [33] and then place and route using Cadence Innovus Implementation System v16.1 [31]. We set the placement density at the floorplan stage as 70%, and perform timing and power analyses using Cadence Innovus Implementation System v16.1. We also validate our optimization framework on two industrial designs, designated as TC1 and TC2, in a 40-nm CMOS foundry technology with high threshold voltage (HVT)-only cells. TC1 contains a dual-core MCU and six memories. TC2 contains more than ten memories and a number of IP blocks. The worst case timing and power analysis views for these two industrial designs are (SS, 0.99 V, −40 °C) and (TT, 1.1 V, 25 °C), respectively. We implement these two industrial designs with Cadence tools. The shifter propagation delay for nominal process, voltage, and temperature (TT, 1.1 V, 25 °C) is 400 ps. For the nonindustrial benchmarks, we generate level

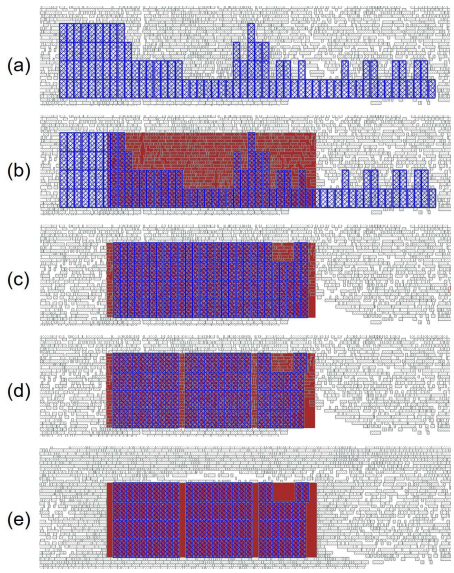


Fig. 10. Example of level shifter insertion. (a) Level shifter (blue) placement after first matching. (b) Placement blockage (red) insertion. (c) Level shifter placement after second matching. (d) Clumping of level shifters. (e) Placement legalization applied to nearby standard cells.

TABLE II
Test-Case Parameters

design	technology	#instances	#flops	clock period
AES	28nm LP	∼11K	530	1.2ns
DES	28nm LP	∼17K	530	1.4ns
JPEG	28nm LP	∼42K	4512	1.6ns
VGA	28nm LP	∼58K	17053	2.0ns
TC1	40nm	∼106K	15245	20ns
TC2	40nm	1.00	0.18	20ns

[7]The instance and flip-flop counts of design TC2 are normalized with respect to the corresponding (instance and flip-flop) counts of the conventional implementation.

TABLE III
EXPERIMENTAL RESULTS (POWER UNIT: mW. CURRENT UNIT: mA. η VALUES ARE ESTIMATED BASED ON [4])

design	flow	WNS (ps)	#inst	inst area (μm^2)	#LS	func mode				sleep mode				runtime (min)
						P_{core}	I_{bot}/I_{top}	P_{ext}	T	P_{core}	I_{bot}/I_{top}	P_{ext}	T	
AES (28nm)	*ref*	-2	10753	8853	0	9.36	9.85/0.00	11.68	1.00	0.08	0.09/0.00	0.55	1.00	-
	opt	-5	11116	11035	169	10.23	**5.37/5.40**	10.40	**1.12**	0.10	**0.06/0.05**	0.16	**3.34**	8
DES (28nm)	*ref*	10	16522	16828	0	17.63	18.56/0.00	22.01	1.00	0.07	0.07/0.00	0.44	1.00	-
	opt	16	16672	18428	135	17.87	**9.86/8.95**	18.39	**1.20**	0.07	**0.04/0.03**	0.09	**4.97**	7
JPEG (28nm)	*ref*	2	42068	47507	0	42.63	44.87/0.00	53.22	1.00	0.29	0.31/0.00	0.72	1.00	-
	opt	5	42737	55376	687	43.36	**21.32/24.33**	44.10	**1.21**	0.31	**0.16/0.16**	0.33	**2.21**	14
VGA (28nm)	*ref*	-1	57969	98004	0	64.82	68.24/0.00	80.92	1.00	0.36	0.38/0.00	0.88	1.00	-
	opt	2	58373	104087	521	65.07	**35.22/33.27**	65.69	**1.23**	0.36	**0.20/0.18**	0.50	**1.75**	21
TC1	*ref*	9	105162	843961	0	13.76	12.51/0.00	17.18	1.00	0.168	0.152/0.000	0.641	1.00	-
	opt	2	106369	861128	601	14.02	**6.17/6.58**	14.54	**1.18**	0.167	**0.075/0.077**	0.175	**3.66**	34
TC2	*ref*	102	1.00	1.00	0	1.00	1.00/0.00	1.00	1.00	1.00	1.00/0.00	1.00	1.00	-
	opt	-36	1.02	1.10	914	1.05	**0.63/0.43**	0.87	**1.15**	1.17	**0.65/0.49**	0.32	**3.17**	113

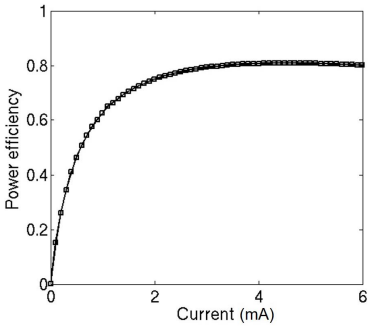


Fig. 11. Power efficiency of switched-capacitor VR used in [4].

shifter models in the 28-nm LP technology according to the delay, area, and power ratios between the level shifter and the minimum-size inverter in the 40-nm technology. Fig. 11 shows the relation between output current versus the power efficiency of the used VR. We use the command *ccopt_design* from Innovus to perform clock tree synthesis. We observe that the tool inserts additional clock buffers to compensate the delay of level shifters for a given skew target. Since our optimization minimizes the number of level shifter insertions, we observe from our experimental results that the power penalty from such additional clock buffer insertion is small (e.g., <1% clock buffer area penalty on design TC2). Our optimization flow is implemented in C++. Functions used in P&R tools are implemented in Tcl. We conduct our experiments using a 2.5-GHz Intel Xeon server.

A. Comparison to Conventional Designs

Table III shows the post-clock tree synthesis (CTS) comparison between our stacked-domain optimization (*opt*) versus the conventional implementation (*ref*) on four test cases in 28-nm LP and two industrial designs in 40 nm. Our optimization comprehends both function mode and sleep mode (i.e., with only leakage power). For the industrial design TC2, we only show normalized metrics—the instance count, area, power (P_{core}, P_{ext}), and battery lifetime (T) values of the optimized designs are normalized to those of the conventional design; the currents (I_{bot}, I_{top}) of the optimized designs are normalized to total current of the conventional design. We estimate η values based on [4].

In 28-nm technology, our optimization achieves an average of 19% and 207% battery lifetime improvements in function and sleep modes, respectively. For the industrial designs in 40 nm, we also achieve more than 10% and 2× battery lifetime improvements in function and sleep modes, respectively. In other words, we observe similar benefits from the stacked-domain optimization in both 28- and 40-nm technologies. Moreover, the power penalty due to our optimization (see P_{core}) is less than 10%, with well-balanced currents (i.e., with <10% difference) between the top and bottom power domains (see $I_{\text{bot}}/I_{\text{top}}$) for most cases. As a result, our optimization significantly reduces P_{ext}, and leads to an improved battery lifetime. We also observe that the battery lifetime increase is greater in the sleep mode. This is because most of the current (leakage) goes through the stacked domains, while the regulator needs to provide very little current to maintain the mid node voltage. Hence, there is a high power delivery efficiency despite the VR having lower efficiency with smaller current (as shown in Fig. 11). Therefore, stacked-domain optimization is expected to provide more energy and battery lifetime benefits if the VR efficiency is low.

The smaller battery lifetime improvement compared with that in [4] is mainly due to placement blockage insertions within level shifter rows (for decap cell insertion) and the constraint of having an even number of level shifter rows (for deep n-well sharing), which incur placement perturbation along with increased design power and ΔI. Furthermore, although the clock buffer area penalty is negligible in our stacked-domain designs compared with the corresponding conventional designs, unbalanced clock power across two domains can increase ΔI and thus reduce battery lifetime benefits.

We note that all the implementation solutions have negligible timing violations (i.e., #timing violation paths <5), and that the slightly improved worst negative slack (WNS) values of our optimization solutions might be due to P&R tools' noise [15]. Moreover, since logic gates are densely connected in blocks AES, DES, JPEG, and VGA, and since the block sizes are small, the relative area overheads due to level shifter insertion are large. Runtimes shown in Table III indicate the

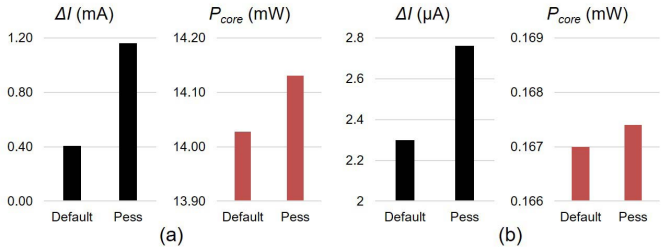


Fig. 12. Impact of level shifter delay, area, and power on design quality of results in (a) function mode and (b) sleep mode. Design: TC1. Technology: 40 nm.

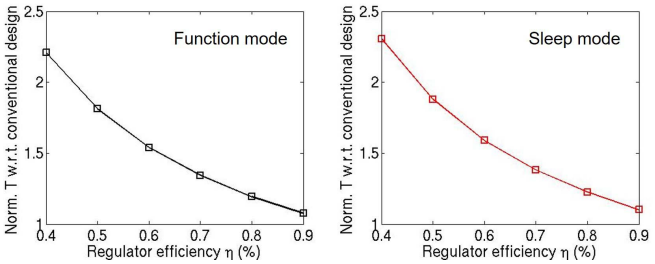


Fig. 13. Impact of VR efficiency on battery lifetime improvement. Design: TC1. Technology: 40 nm.

extra runtime of our optimization that includes partitioning, refloor planning, and incremental placement optimization.

B. Sensitivity to Level Shifter Delay

We further study the impact of level shifter model on our stacked-domain optimization. We use a pessimistic (i.e., worst case) model that has roughly 3–4× power, area, and delay compared our current (i.e., nominal-case) model. Fig. 12 shows that the pessimistic level shifter model leads to slightly larger total design power (P_{core}) due to larger level shifter power and timing impact. Since our partitioning optimization minimizes the number of level shifters, the corresponding power penalty due to the pessimistic level shifter model is not large. Results also show larger ΔI with the pessimistic level shifter model. The larger current difference comes from the level shifters' timing and area impact.

C. Sensitivity to Voltage Regulator Power Efficiency

We study the impact of VR efficiency on battery lifetime improvement in stacked-domain designs. More specifically, we vary the η value from 40% to 90% with a step size of 10%. For each η value, we estimate the battery lifetime improvement from our stacked-domain optimization compared with the conventional design. Figs. 13 and 14, respectively, show normalized battery lifetime with respect to that of the conventional design, evaluated using different VR efficiencies. Results show that battery lifetime decreases with a higher VR efficiency. When the regulator efficiency is high, stacked-domain implementation—which has power penalty due to level shifter insertion—can even degrade the battery lifetime of the design (e.g., $\eta = 90\%$ in sleep mode).

We note that the VR efficiency and area cannot both be optimal in the same power converter [23]. In a typical CMOS process, while highly efficient converters exist, they have poor power density, and the opposite holds for high power density

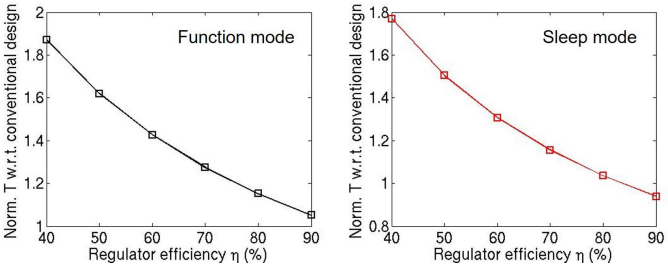


Fig. 14. Impact of VR efficiency on battery lifetime improvement. Design: TC2. Technology: 40 nm.

converters, where the efficiency is reduced. Since optimization of both power efficiency and power density cannot be performed beyond technology limitations, an alternate method for further improvement is voltage stacking. Here, the better the matching between the power domains, the less current the regulator has to provide, improving the maximum output current requirements that are related to the area, and also improving the external power supply and, therefore, the power delivery efficiency, in accordance with (5).

D. Tradeoff Between Current Balancing Versus Level Shifter Cost

A relaxed current balancing constraint will lead to a smaller number of level shifters, and hence reduced area and power penalties. In this section, we study this tradeoff between current balancing versus the cost of level shifters (e.g., the number of level shifters and resultant P_{core} increase) using the industrial designs TC1 and TC2. Specifically, we vary the instances defined as source and sink as well as the maximum delta current constraint in our flow-based partitioning to achieve different partitioning solutions with different numbers of level shifters.

Table IV shows results for designs TC1 and TC2 with different current balancing constraints, normalized to those for conventional designs. In Table IV, each column corresponds to one implementation—from left to right, ΔI increases while the number of level shifter insertions decreases. Specifically, the leftmost column (*ref*) corresponds to the conventional design with only one power domain. The second column from the left (*opt'*) indicates the stacked-domain implementation with a small number of level shifters and unbalanced currents. The rightmost column (*opt*) corresponds to the stacked-domain implementation with well-balanced currents but a large number of level shifters. We then evaluate battery lifetime of each implementation with three VR efficiency assumptions (i.e., $\eta = 50\%$, 80%, and 95%). In Table IV, for each η value (i.e., for each row), the solution with the maximum battery lifetime is shown in bold font.

Results in Table IV show that when η is small, solutions with more balanced current offer larger battery lifetime. On the other hand, when η is large, solutions with relaxed current balancing constraints and a smaller number of level shifters provide larger battery lifetime (e.g., on design TC2, when $\eta = 95\%$, *opt'* compared with other implementations). This is because when η is large, ΔI does not have large impact

TABLE IV

RESULTS WITH DIFFERENT CURRENT BALANCING CONSTRAINTS. DESIGNS: TC1 AND TC2. ΔI, P_{core}, AND T ARE NORMALIZED TO THOSE OF THE CONVENTIONAL DESIGN

design	metric	*ref*	*opt'*	*opt"*	*opt*
TC1	ΔI	1.00	0.91	0.88	0.03
	P_{core}	1.00	1.00	1.01	1.02
	#LS	0	174	406	601
	T (η = 50%)	1.00	1.11	1.32	**1.62**
	T (η = 80%)	1.00	1.05	1.09	**1.15**
	T (η = 95%)	1.00	1.01	1.00	**1.03**
TC2	ΔI	1.00	0.82	0.50	0.20
	P_{core}	1.00	1.00	1.02	1.05
	#LS	0	362	513	914
	T (η = 50%)	1.00	1.06	1.07	**1.91**
	T (η = 80%)	1.00	1.02	1.02	**1.22**
	T (η = 95%)	1.00	**1.02**	1.01	1.01

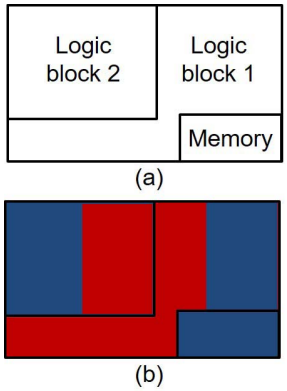


Fig. 15. (a) Approximate layout of TC2. (b) Approximate partitioning solution of TC2 (red: top domains and blue: bottom domains).

on battery lifetime. Moreover, with a relaxed ΔI constraint, the number of level shifter insertions and corresponding power penalty are small. Results in Table IV also show that since our optimization is able to achieve a balanced-current partitioning solution with a small number of level shifter insertions, for design TC1, our solution (*opt*) achieves the maximum battery lifetime for various VR efficiencies.

E. Block-Aware Partitioning

Design TC2 has two usage scenarios—with and without logic block 2. To ensure that the currents between two domains are balanced in both scenarios, we perform block-aware partitioning on TC2. To achieve this, we first partition logic block 1 and memories into two domains with balanced currents. We then fix the partitioning solution on logic block 1 and memories, and partition logic block 2 into two domains with the current heuristically balanced for the entire design, and with a minimized number of level shifter insertions. Fig. 15(a) shows the relative locations of three blocks. Fig. 15(b) shows the relative locations of top (red) and bottom (blue) domains within each block. We are unfortunately not able to provide additional floor plan and layout details for TC1 and TC2. We note that since we perform partitioning optimization based on trial placement, which has been optimized to minimize

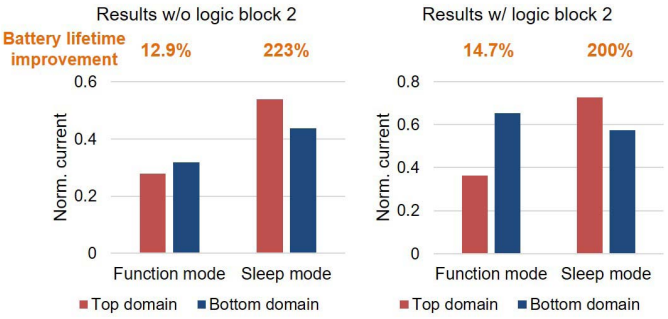


Fig. 16. Block-aware partitioning solution, evaluated in both scenarios (with and without logic block 2). Current values are normalized to the total current of the conventional design including both logic block 1 and logic block 2. Battery lifetime improvements are with respect to the conventional design.

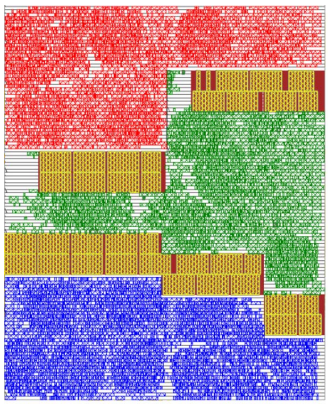
the wirelength, and since our level shifter locations are determined by the matching optimization, which minimizes the wirelength, our optimized design does not have large overhead in signal wirelength. However, splitting the bottom domain into two regions increases the complexity of the VR and implied control circuitry, and incurs routing overhead between the power management unit (PMU) and the bottom domain as well as area overhead due to more local power switches. We have also explored other partitioning solutions having a contiguous region for each power domain in an attempt to reduce the overheads in PDN, but all alternatives studied result in a large number of level shifters.

Fig. 16 shows the optimized solution of TC2, evaluated in the two scenarios of with and without logic block 2 working. The VR efficiency is estimated based on the power delivery block described in [4]. The solution has negligible timing violations (i.e., the WNS is -60 ps, with less than five path timing violations) and 1987 level shifters. Results show that by applying the block-aware partitioning, we achieve balanced currents in both working scenarios, and thus improved battery lifetime. The currents are not perfectly balanced because of the high complexity of the industrial design (otherwise, there will be power and timing penalties from the large number of level shifter insertions) as well as the multiscenario (i.e., function mode and sleep mode) balancing constraints.

F. Partitioning With Multiple Power Domains

We further validate our partitioning optimization with multiple power domains. Fig. 17 shows our partitioning result on design AES with three power domains. Specifically, we perform two flow-based partitioning optimizations in a sequential way. The first partitioning optimization splits instances into two parts with a current ratio of 2:1. The second partitioning optimization then divides the larger-current part into two domains with balanced currents. Last, we insert level shifters based on the partitioning solution and perform matching optimization to determine level shifter locations. Results in Fig. 17 show that currents are balanced across three domains.

Although our proposed flow is able to address stacked-domain designs with multiple power domains, a stacked-domain design with more than two power domains might



Experimental results.

WNS (ps)	5
#inst	11186
inst area (μm^2)	11578
#LS	321
func mode P_{core} (mW) $I_{top}/I_{mid}/I_{bot}$	10.46 32%/39%/29%
sleep mode P_{core} (mW) $I_{top}/I_{mid}/I_{bot}$	0.106 30%/34%/36%

Fig. 17. Partitioning result with three power domains. Blue, green, and red are, respectively, instances assigned to the bottom $\{V, 0\}$, middle $\{2V, V\}$, and top $\{3V, 2V\}$ domains. Yellow: level shifters. Design: AES.

face several limitations, as follows: 1) partitioning with more than two power domains typically leads to more number of cuts, where the increased number of level shifters incur power and area overheads; 2) stacked-domain designs with more than two power domains demand a more complex power delivery topology and VRs; and 3) stacked-domain designs with more than two power domains might require additional types of level shifters if the VDD range is large (e.g., level shifters between domains $\{V, 0\}$ and $\{3V, 2V\}$).[8] Finally, we note that for particular VRs (e.g., switched-cap converter), 2:1 conversion ratio (applied in a stacked-domain design with two domains) has higher efficiency compared with other conversion ratios (e.g., 3:1 and 3:2 in a stacked-domain design with three power domains). The lower conversion ratio incurs power penalty when currents are not perfectly balanced among power domains.

V. Conclusion

In this paper, we propose the first comprehensive optimization framework for stacked power-domain implementation with maximized battery lifetime. We extend the existing flow-based partitioning methodology with layout- and timing-path awareness, as well as multiscenario balancing objective. We further propose an FM-based bin movement and a dynamic programming-based boundary optimization to define the layout region (power island) of each power domain. Last, we insert level shifter rows in an updated floor plan and place level shifters using a matching optimization. We validate our optimization flow in both 28-nm LP and 40-nm technologies, as well as on industrial designs. Our optimization achieves more than 10% and 2× battery lifetime improvements for function and sleep modes compared with the conventional design. Our future works include: 1) a predictive methodology to determine the block size prior to trial placement and 2) co-optimization of stacked-domain partitioning and floorplan update.

[8]In our experiments, we cascade two level shifters to connect domains $\{V, 0\}$ and $\{3V, 2V\}$. However, this will increase the number of level shifter insertions.

References

[1] C. J. Alpert and A. B. Kahng, "Recent directions in netlist partitioning: A survey," *Integr. VLSI J.*, vol. 19, nos. 1–2, pp. 1–81, 1995.

[2] K. Blutman, H. Fatemi, A. B. Kahng, A. Kapoor, J. Li, and J. P. de Gyvez, "Floorplan and placement methodology for improved energy reduction in stacked power-domain design," in *Proc. ASP-DAC*, 2017, pp. 444–449.

[3] K. Blutman, A. Kapoor, J. G. Martinez, H. Fatemi, and J. P. de Gyvez, "Lower power by voltage stacking: A fine-grained system design approach," in *Proc. DAC*, 2016, pp. 78-1–78-5.

[4] K. Blutman *et al.*, "A microcontroller with 96% power-conversion efficiency using stacked voltage domains," in *Proc. IEEE Symp. VLSI Circuits*, Jun. 2016, pp. 1–2.

[5] A. C. Cabe, Z. Qi, and M. R. Stan, "Stacking SRAM banks for ultra low power standby mode operation," in *Proc. DAC*, 2010, pp. 699–704.

[6] A. E. Caldwell, A. B. Kahng, and I. L. Markov, "Improved Algorithms for Hypergraph Bipartitioning," in *Proc. ASP-DAC*, 2000, pp. 661–666.

[7] A. E. Caldwell, A. B. Kahng, and I. L. Markov, "Optimal partitioners and end-case placers for standard-cell layout," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 19, no. 11, pp. 1304–1313, Nov. 2000.

[8] 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.

[9] S. Devadas and S. Malik, "A survey of optimization techniques targeting low power VLSI circuits," in *Proc. DAC*, 1995, pp. 242–247.

[10] S. Dutt and W. Deng, "VLSI circuit partitioning by cluster-removal using iterative improvement techniques," in *Proc. ICCAD*, 1996, pp. 194–200.

[11] C. M. Fiduccia and R. M. Mattheyses, "A linear time heuristic for improving network partitions," in *Proc. DAC*, 1982, pp. 175–181.

[12] M. X. Goemans and D. P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *J. ACM*, vol. 42, no. 6, pp. 1115–1145, 1995.

[13] L. Guo, Y. Cai, Q. Zhou, and X. Hong, "Logic and layout aware voltage Island generation for low power design," in *Proc. ASP-DAC*, 2007, pp. 666–671.

[14] L. W. Hagen, D. J.-H. Huang, and A. B. Kahng, "On implementation choices for iterative improvement partitioning algorithms," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 16, no. 10, pp. 1199–1205, Oct. 1997.

[15] K. Jeong and A. B. Kahng, "Methodology from chaos in IC implementation," in *Proc. ISQED*, 2010, pp. 885–892.

[16] A. B. Kahng and X. Xu, "Local unidirectional bias for smooth cutsize-delay tradeoff in performance-driven bipartitioning," in *Proc. ISPD*, 2003, pp. 81–86.

[17] G. Karypis and V. Kumar, "Multilevel K-way hypergraph partitioning," in *Proc. DAC*, 1999, pp. 343–348.

[18] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 359–392, 1998.

[19] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell Syst. Tech. J.* vol. 49, no. 2, pp. 291–307, 1970.

[20] B. Krishnamurthy, "An improved min-cut algonthm for partitioning VLSI networks," *IEEE Trans. Comput.*, vol. C-33, no. 5, pp. 438–446, May 1984.

[21] S. K. Lee, T. Tong, X. Zhang, D. Brooks, and G.-Y. Wei, "A 16-core voltage-stacked system with an integrated switched-capacitor DC-DC converter," in *Proc. Symp. VLSI Circuits*, 2015, pp. C318–C319.

[22] Y. Liu *et al.*, "A 0.1 pJ/b 5-to-10 Gb/s charge-recycling stacked low-power I/O for on-chip signaling in 45 nm CMOS SOI," in *Proc. ISSCC*, 2013, pp. 400–401.

[23] M. Steyaert *et al.*, *DCDC Performance Survey*. Accessed on Jan. 6, 2017. [Online]. Available: http://homes.esat.kuleuven.be/~steyaert/DCDC_Survey/DCDC_PS.html

[24] S. Rajapandian, K. Shepard, P. Hazucha, and T. Karnik, "High-tension power delivery: Operating 0.18 μm CMOS digital logic at 5.4 V," in *Proc. ISSCC*, 2005, pp. 298–299, and 599.

[25] R. Rajaraman and D. F. Wong, "Optimum clustering for delay minimization," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 14, no. 12, pp. 1490–1495, Dec. 1995.

[26] M. Shih and E. S. Kuh, "Quadratic Boolean programming for performance-driven system partitioning," in *Proc. DAC*, 1993, pp. 761–765.

[27] K. Ueda, F. Morishita, S. Okura, L. Okamura, T. Yoshihara, and K. Arimoto, "Low-power on-chip charge-recycling DC-DC conversion circuit and system," *IEEE J. Solid-State Circuits*, vol. 48, no. 11, pp. 2608–2617, Nov. 2013.
[28] H. Wu, I.-M. Liu, M. D. F. Wong, and Y. Wang, "Post-placement voltage island generation under performance requirement," in *Proc. ICCAD*, 2005, pp. 309–316.
[29] H. H. Yang and D. F. Wong, "Efficient network flow based min-cut balanced partitioning," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 15, no. 12, pp. 1533–1540, Dec. 1996.
[30] *Hungarian Algorithm*. Accessed on Oct. 17, 2016. [Online]. Available: http://www2.informatik.uni-freiburg.de/~stachnis/misc.html
[31] *Cadence Innovus User Guide*, Cadence Design Systems, San Jose, CA, USA, 2016.
[32] OpenCores. Accessed on Sep. 8, 2015. [Online]. Available: http://opencores.org
[33] *Design Compiler User's Manual*, Synopsys, Mountain View, CA, USA, 2013.

Kristof Blutman received the M.Sc. degree (Hons.) in electrical engineering from the Delft University of Technology, Delft, The Netherlands, in 2014.

In 2014, he joined NXP Semiconductors, Eindhoven, The Netherlands, as a Senior Scientist, where he is involved in low-power techniques for microcontrollers. His current research interests include the fields of energy-efficient digital systems and power conversion methods.

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 currently an Innovation Lead/Department Manager with NXP Semiconductors, Eindhoven. He has authored or co-authored over 25 U.S. patents, scientific publications, and presentations. His current research interests include the areas of low-power design, multiprocessors, heterogeneous and reconfigurable systems, and variability tolerance design.

Ajay Kapoor received the B.Tech. degree in electrical engineering from IIT Delhi, New Delhi, India, and the M.Sc. (*cum laude*) degree in embedded systems from the University of Twente, Enschede, The Netherlands.

Since 1999, he has been with NXP Semiconductors, Eindhoven, The Netherlands, where he has been involved in the design of low-power circuits, system, and algorithms. His current research interests include low-power circuits, architectures, and signal processing.

Andrew B. Kahng received the Ph.D. degree in computer science from the University of California at San Diego, La Jolla, CA, USA.

He is currently 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 road mapping.

Jiajia Li received the B.S. degree in software engineering from Shenzhen University, Shenzhen, China, in 2011 and the M.S. degree in electrical engineering from the University of California at San Diego, La Jolla, CA, USA, in 2013, where he is currently pursuing the Ph.D. degree.

He joined the VLSI CAD Laboratory, University of California at San Diego, in 2012. His current research interests include physical design and signoff optimization, margin reduction, and low-power design.

José Pineda de Gyvez was a Faculty Member with the Department of Electrical Engineering, Texas A&M University, College Station, TX, USA. He is currently a fellow with NXP Semiconductors, Eindhoven, The Netherlands, 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, Eindhoven. This professorship fills a gap between industry and academia by bringing industrial knowledge into classrooms, and open innovation into NXP. He has over 150 publications in the fields of low-power IC design, analog signal processing, and design for manufacturability and test. He has (co)-authored four books. He has over 20 U.S. granted patents.

Dr. Pineda de Gyvez has been an associate editor of several IEEE transactions. He is often involved in program and steering committees of international symposiums. He is a member of the Editorial Board of the *Journal of Low Power Electronics*.