**Title**

Good morning. Thank you for being my committee professors. The title of my thesis and this presentation is improved physical design and signoff methodologies for better integrated circuit design quality.

**Slide**

This slide shows the outline of my thesis. There are four chapters in my thesis. The first chapter gives an overview. The following three chapters present three optimizations to address three major challenges in physical design and signoff.

(Q: what is physical design and what is signoff ?)

The three major challenges are complex operating conditions, demand for low-power designs and growing design margins. And my three chapters are multi-mode multi-corner optimization, low-power optimization and mixed-fabric optimization. I will give details regarding the challenges and the proposed optimizations in the following slides.

**Slide**

This slide shows my publications during my Ph.D. study.

**Slide**

Now I will discuss some motivation and more details about my thesis. As noted by ITRS, the realizable transistor density scaling in actual MPU products has slowed down from traditional 2X per technology node to 1.6X per technology node.

The figure shows that there is a design capability gap between the available scaling and realizable scaling.

(Q: what is density? what is available scaling? what is realizable scaling? MPU?)

To compensate the design capability gap, we pursue design-based equivalent scaling, that is, to rely on design technology improvement to achieve performance, power, area and cost tradeoffs to rescue Moore’s-Law scaling of value.

As key steps in IC design, physical design and signoff need to be improved to achieve design-based equivalent scaling.

Moreover, there are physical design and signoff also face many challenges.

**Slide**

This slide summarizes the major challenges in physical design and signoff.

The first challenge is the complex operating conditions and corner explosion. High-performance and low-power designs typically have multiple operating modes such as turbo mode and nominal mode. The multi-mode operation requires multi-mode signoff.

In addition, there is test mode, of which the timing and power can also be critical. Test mode optimization must no degrade QoR in function mode.

Further, there is ping-pong effect during multi-mode optimization, that is, optimization at one mode can cause timing violations in other modes.

So, we see that multi-mode multi-corner optimization is a challenge.

The second challenge is the demand for low-power designs.

Power reduction has been viewed as a grand challenge in ITRS. Moreover, low-power techniques will increase design complexity and introduce power overheads. Therefore, we must ensure that the power benefits from these techniques outweigh their costs. And since there are urgent requests for low-power designs, aside from commonly used techniques, new techniques need to be proposed.

(Q: list low-power techniques)

Last challenge is the growing design margins. Due to increased design complexity, process variation and reliability constraints, designers use overdesign to ensure correctness of the design. However, such margins will reduce potential benefits from technology scaling.

(Q: what is margin?)

**Slide**

To pursue design-based equivalent scaling and address the challenges for better design QoR, my thesis presents improve physical design and signoff methodologies.

This figure shows the scope of my thesis. As discussed, three chapters address three major challenges. And details of each chapter is also described.

**Slide**

In this presentation, I will present three works. One from the multi-mode and muti-corner optimization chapter, two from low-power optimization chapter.

**Slide**

The first work is comprehensive optimization of scan chain timing during late-stage IC implementation. This is a joint work with Samsung.

**Slide**

This slide gives a brief introduction on scan chain.

Scan chain technique is commonly used in design for test. It provides a simple way to set and observe all flip-flops. The bottom figure shows an example. In the example, there are three scan flip-flops. They form a scan chain.

There are three stages during scan test. During the scan in stage, we shift in and load all flip-flops with an input vector. Then the capture stage excites combinatorial logic and capture outputs at flip-flops. Finally, during the scan out phase, we shift out the captured output vector and check whether there is an error.

The scan in and scan out together is called scan shift stage, which takes much test time due to large number of cycles. In this work, we optimize scan shift timing.

**Slide**

Scan timing, especially scan shift timing, is important to test time, test cost and test robustness.

In this talk, I will discuss two scan timing issues.

First, since the number of logic instances along a scan timing path is typically small, scan timing paths are vulnerable to hold violations. As a result, many hold buffers are inserted, which increase design area and routing congestion.

Second, scan shift is typically performed at a high frequency. The corresponding high power will incur large dynamic voltage drop. And the large DVD further degrades scan timing and leads to “false failure” during test.

To address these two issues. Our goals in the work are to perform scan ordering for hold buffer reduction and to insert gating logics to minimize timing degradation due to dynamic voltage drop.

These problems are not new. But do previous approaches really solve these problems?

**Slide**

Most of previous approaches optimize scan chain during early design stages, such as synthesis and placement.

However, the hold-critical paths and DVD hotspots can very between early and late design stages.

This figure shows that the hold-critical scan timing paths, which are in red, change between the post-placement and post-routing stages. The difference might come from the impact of clock skew and interconnect delay.

This figure shows large difference of DVD map between the post-placement and the post-routing stages. The difference might come from the clock buffers and timing optimizations during routing, such as sizing and buffering.

Since the hold timing and dynamic voltage drop vary between early and late design stages, an early-stage scan chain optimization might be misleading. We therefore propose optimizations during late-stage IC implementation.

However, the late-stage optimization is not trivial. It has to consider the timing impact on datapaths in function mode. It also needs to minimize the area and power overheads to avoid design QoR degradation.

**Slide**

Now, I will first describe our methodology for scan ordering for hold buffer removal.

We define the post-routing scan ordering problem as Given a routed design, timing constraints and upper bound on wirelength penalty, we perform scan ordering to minimize the number of hold buffers.

**Slide**

Before descrbing our methodology for scan ordering, we first study the causes of hold violations on scan timing paths.

This figure shows the skew distribution of scan timing paths with hold buffers. We see that majority of hold-critical paths has negative skew values.

This figure shows distances between the launch and capture flip-flops versus their hold slacks. We see that smaller distances lead to smaller hold slacks.

**Slide**

Based on these observations, we perform optimizations to achieve greater incidence of positive skew values and slightly increase start-end flip-flop distances, to remove hold buffers.

**Slide**

This slide shows the pseudo code of our scan ordering optimization. We iteratively perform two-opt optimizations along the scan chain and update the ordering solution with the one with smaller number of hold buffers.

The right figure shows an example of scan ordering to exploit clock skew for hold buffer removal.

In our scan ordering optimization, we do not allow timing degradation on datapaths and additional hold violations. And our optimization always meets a predefined upper bound on wirelength penalty. To honor a fixed scan chain ordering, a subchain with fixed ordering is merged into one node before our optimization.

**Slide**

Now I will describe our methodology of DVD-aware gating insertion. We define the problem as Given a routed design, timing constraints, power information and upper bound on area overhead, we perform gating insertion to maximize the minimum DVD-aware slack.

**Slide**

This slide shows our overall gating insertion flow. We first determine the DVD hotspots to optimize. Here a DVD hotspot is a grid with large dynamic voltage drop. Note that we only consider the DVD hotspots having impact on scan timing slacks. We also note that the worst DVD hotspot is not necessarily the same as the hotspot with the largest timing impact.

Second, we find gating locations within a netlist to reduce the dynamic power within the selected DVD hotspots. We also minimize the number of gating logic insertions to minimize the area overhead.

Last, we perform ECO-based gating insertion.

The figure shows the schematic of gating insertion. Note that we also insert gating logics inside the logic cone.

I will discuss the details of each step in the following slides.

**Slide**

We formulate an integer linear program to select DVD hotspots We select a limited number of DVD hotspots to optimize so as to maximize the minimum DVD-aware slack.

Our ILP formulation is shown here. As mentioned, we maximize the minimum DVD-aware slack. The first constraint estimates the slack improvements from DVD reductions within the selected DVD hotspots. The second constraint ensures that a DVD hotspot is selected if any cell within it is gated. The third constraint enforces an upper bound on the number of the selected DVD hotspots.

**Slide**

We perform netlist traversal to find gating locations. Our objective here is to minimize the dynamic power within selected DVD hotspots. A simply example is shown in the figure. In the example, in red are cells within the selected hotspots and in white are candidate gating locations. We first assign a gain of one to each cell within the selected hotspots. We then propagate the gain values from each cell within the selected hotspots backwards based on the number of fanins. Last, we select the gating location with the maximum gain value.

In the last step of our optimization, we perform a matching optimization between available white spaces and gating logics and insert gating logics as ECO steps.

**Slide**

This slide shows our experimental setup. We perform experiments in 28LP technology. The tools we use are shown here. We use four designs shown in the table as our testcases.

**Slide**

This slide shows our scan ordering results. We use the default SP&R flow based on commercial tools as our reference flow. We see from the table that our optimization achieves 82% hold buffer reduction and with very small wirelength penalties.

**Slide**

This slide shows our gating insertion results. Again, we use the default SP&R flow as our reference flow. We see from the table that our optimization achieves up to 58% improvement of DVD-induced slack degradation. Our optimization achieves this with small number of gating logics. Therefore the area penalty is small. We also see that the worst DVD is not significantly optimized. This indicates that the worst DVD does not necessarily correspond to the worst DVD-aware slack.

**Slide**

We propose comprehensive scan timing optimization during late-stage IC implementation. We validate our optimization with a realistic implementation flow. Our optimization leads to up to 82% hold buffer reduction and up to 58% improvement of DVD-induced scan timing degradation.

Our future works are listed as follows.

SLIDE

The second topic for today’s presentation is improved flop tray-based design implementation for power reduction.

SLIDE

First, a flop tray here is a multi-bit flop-flop, which is a combination of flip-flops

We know that the application of flop trays can significantly reduce the number of sinks in a clock tree, thus reducing clock tree wirelength and clock power.

As a simple calculation, in a given clock tree, if we replace all the single-bit flops with 64-bit flop trays, we are able to reduce the number of clock buffers by 98%.

Further, if we assume a clock tree has 100K sinks and fanout of eight at each level, by replacing all the single-bit flops with 64-bit flop trays, we can reduce the clock tree depth from six to four.

From these examples, we can see that the usage flop trays can significantly reduce the number of clock buffers and clock power.

I will show additional benefits from flop trays in the next slide.

SLIDE

This figure shows a single-bit flip-flop.

We see that each flop generates its own clock signals with two inverters.

In a flop tray, the inverters for clock signals can be shared, which reduces power and area of flops.

As an example, a recent work achieves 22% flop power reduction by using 2-bit and 4-bit flop trays.

So, we see that the usage of flop trays not only reduces clock power but also power and area of flop itself.

However, implementation of flop tray-based design is not trivial. I will show several challenges in the following slides.

SLIDE

First, flops occupy large portion of block area. As an example, in one of our testcases, VGA, 30% of instances are flops, which takes 51% of the block area. Therefore, the optimization of flops and flop trays has significant impact on design quality.

Second, flop trays can have high aspect ratio and distinct size. As shown in the bottom figure, the 64-bit flop trays, which are in orange, have very high aspect ratio. Unable to comprehend the dimension of flop trays will result in degraded solution quality.

Third, clustering of flops for flop tray generation imposes additional placement constraints, which can easily increase routing congestion and power penalty.

So we see that there is a tradeoff between flop tray benefits versus datapath power penalty. In other words, if we only use small flop trays, we cannot fully exploit the benefit of flop trays. On the other hand, using large-size flop trays may sacrifice datapath wirelength and power.

This figure shows power and wirelength overheads on datapath from a logical clustering flow, where we use commercial tools to cluster flops during the synthesis stage. We can see up to 40% increase in wirelength, and up to 16% increase in datapath power due to flop tray generation.

Therefore, we must ensure that the benefits of using flop tray outweigh its costs.

SLIDE

This slide shows our overall optimization flow. In blue are our proposed optimizations. We first synthesize the netlist with only single-bit flip-flops. We then place the synthesized netlist with only single-bit flops. We consider such an initial placement solution as a relatively good placement solution in terms of routing congestion and datapath timing and power, since there is no additional placement constraint from flop clustering.

Based on the initial placement solution, we perform flop clustering and generate flop trays. Our objective here is to minimize the displacement of flops as well as timing and power impact on datapath. These indicate that we want to maintain the solution quality of the initial placement as much as possible. At the same time, we also minimize the number of flop trays to reduce clock power and power of flops.

Our separate study shows that it is practically impossible to optimally optimize flop clustering and flop tray placement simultaneously with all possible flop tray sizes.

We therefore perform a two-step optimization. The first step is a capacitated K-means clustering, as shown in the dotted box, in which we generate optimal clustering solutions for each flop tray size. We then perform ILP-based selection to generate a flop tray solution with mixed flop tray sizes.

Based on the generated flop trays, we then perform placement legalization, clock tree synthesis and routing.

I will show an example in the next slide.

SLIDE

As discussed, we first generate flop tray solution for each flop tray size. We then combine the solutions.

This slide shows an example of our flow.

The first three figures show the flop tray solutions with 4-bit, 16-bit and 64-bit flop trays. The last figure shows a combined solution from our ILP-based optimization. I will give details of our two-step optimization in the following slides.

SLIDE

This slide describes our capacitated K-means clustering. Specifically, we solve the following problem. Given N points, which are locations of single-bit flops, a capacity K (which is defined by the flop size), we obtain N/K clusters to minimize the total displacement.

Our flow is shown here.

First, we select N/K initial points as centers. In this step, we first randomly select one flop among the single-bit flops. We then calculate the distance from each flop to all selected flops. We use the distance as probabilities to randomly select the next points. We iterative update the probabilities and select points until we select N/K points.

Since the selection of initial points affects final solution quality and there is randomness in our optimization, we use multi-start technique in our optimization to achieve a better solution quality.

Based on the selected centers, our K-means clustering iteratively performs clustering and center location update to generate flop tray solution.

For clustering we formulate a min-cost flow to map single-bit flops to flop tray slots. The figure shows the flow network, where S and T are super source and sink. h are single-bit flops, f are slots on flop trays.

The cost between a single-bit flop and a flop tray slot is the Manhattan distance between them. We note that by considering the distance between flops and slots, we are aware of the flop tray aspect ratios.

To update the center locations, we formulate a linear program. Our LP simply minimizes the total displacements of flops as shown in the formulation.

In our optimization, we iterate between the min-cost flow-based clustering and LP-based center location update until the center movement is negligible.

SLIDE

Here is an example of our optimization. In blue are single-bit flops from an initial placement. In red are the center locations or flop tray locations. We use different colors to indicate different clusters.

We can see that we start with randomly selected N/K centers. We then iteratively update the clustering and center location until the optimization converges.

SLIDE

Recall that our optimization comprehends flop tray shapes by using distance between single-bit flops and slots in flop trays as cost in the clustering optimization.

This slide compares our clustering solution which understands flop tray shapes versus a solution of the traditional K-means clustering which treats each flop tray as a point.

The dots are single-bit flops and shaded rectangles are flop trays. We use different colors to indicate different clusters.

We see that our clustering solution more closely matches the aspect ratio of the flop trays.

SLIDE

Recall that we first generate flop tray solution for each flop tray sizes. We then formulate an integer linear program to combine the solutions with mixed flop tray sizes. This slide shows our ILP formulation.

Our goal here is to minimize total displacement of flops, timing impact due to flop clustering and total flop tray cost such as number of flop trays and flop tray power. Here, W is the total cost of flop trays, which can be estimated based on their area or power, D is the total displacement, and Z is the total relative displacement of timing-critical start-end pairs. We minimize the relative displacement of timing-critical start-end pairs to minimize the timing impact of flop clustering. I will give details in the next slide.

The first two constraints calculate flop displacement.

These two constraints estimate total relative-displacement between timing-critical start-end flop pairs.

This constraint calculates the cost of flop trays, where e is a binary indicator of whether a flop tray is used.

The last constraint ensures that each flop has exactly on slot to match and each slot can have at most one flop to match.

Note that since each single-bit flop only has a limited number of flop trays to match, the runtime of our ILP is small. In our experiments, the runtime is less than one minute of the VGA testcase with 17K flops and five candidate flop tray sizes.

SLIDE

Recall that there is a tradeoff between flop tray benefits versus the datapath power penalty. In our optimization, the choice of alpha in the objective function optimizes such a tradeoff.

The figure shows numbers of flop trays with different sizes and average displacement of each flop vary with the alpha value.

We see that when alpha is small, small-size flop trays are used and the displacement is small. On the other hand, when alpha is large, large-size flop trays are used and the displacement is large.

In our experiments, we use several alpha values in the optimization and select the best outcome.

SLIDE

Recall that we minimize the relative displacement to minimize timing impact.

This slide illustrates our idea on the relative displacement.

For a timing-critical start-end flop pair, relative displacement between them degrades timing. When they are moved apart, the delay will increase due to longer wire. When they are moved closer, there can be congestion in between.

We therefore want to minimize the relative displacement of timing-critical start-end pairs. This figure shows our optimization results with different beta values. When beta is zero, we do not consider relative displacement. When beta is larger, we assign more weight to relative displacement in our objective function. We see from the result that by considering the relative displacement of timing-critical start-end pairs, we achieve 5% more power reduction.

SLIDE

Now I will present our experimental results.

This slide shows our experimental setup.

We perform experiments on four designs from opencores website.

We use foundry 28 FDSOI dual-VT library.

We use Design Compiler for synthesis, and Innovus for physical implementation and analysis.

The bottom table shows our used flop trays. We have five flop tray sizes, ranges from 4-bit to 64-bit. Their normalized power and area as well as the aspect ratios are shown in the table.

SLIDE

This slide shows our experimental results. We compare to two reference flows. Ref\_1b is the conventional implementation flow with only single-bit flops. ref\_mb is the flop tray-based implementation with logical clustering during synthesis. Our optimization is opt\_mb.

We can see that our optimization achieves up to 98% reduction clock tree sink number reduction, and 90% clock power reduction compared to the conventional single-bit flow.

And 16% more total power reduction compared to the flow with logical clustering.

SLIDE

This slide shows the example layouts before and after flop tray generation. In red are flops and in blue are combinational cells. We can see that different sizes of flop trays are used. For design MPEG, we can also see some white space near flop trays. That’s because flop trays are more area efficient than single-bit flops. This effect may also help to reduce routing congestion.

SLIDE

This slide shows optimization with various flop tray sizes. We create five combinations, with different bounds on the largest tray size.

The figure shows clock power with different combinations of flop tray sizes normalized to the clock power with only single-bit flops.

We can see that about 50% of clock power can be reduced by just applying 4-bit flop trays.

With 16-bit or larger flop trays, we can achieve 11% more clock power reduction on average, especially on large designs.

SLIDE

We further study the useful skew optimization with flop trays. Useful skew is helpful to reduce datapath leakage power. However, application of flop trays will limit the benefits from useful skew optimization. We therefore modified our clustering approach to avoid clustering flops with large difference in desired latencies.

More specifically, we calculate the optimal clock latency for each sink to maximize the slacks at each endpoint. We then avoid clustering flops with optimal required latency larger than a particular threshold. This enables skew-awareness.

From the table, we can see that we achieve similar leakage power compared to the solution with only single-bit flops, but at the cost of 21% less sink number reduction compared to our original flop tray-based solution.

SLIDE

Now I will give the conclusion.

In this work, we propose a novel flop tray-based optimization with capacitated K-means algorithm.

We achieve up to 16% total reduction compared to a conventional clustering flow.

We also perform useful skew optimization in the context of flop tray based design.

Our ongoing works include

Scalable optimization considering all flop tray sizes at the same time.

And floorplan blockage awareness.

Slide

My third topic for today’s presentation is floorplan and placement methodology for improved energy reduction in stacked power domain design. This is a joint work with NXP.

**Slide**

Battery lifetime is critical to IC designs, especially for IoT and mobile applications.

But the misalignment between battery and core voltages can cause power inefficiency in voltage regulator.

In this work, our goal is to improve the power delivery efficiency and to improve battery lifetime through stacked-domain optimization.

A stacked-domain design connects two power domains with balanced current in series.

Bottom figure illustrate the difference between a conventional design and a stacked-domain design.

As an example, if the supply voltage of a conventional design is 1V. In the stacked-domain design, the VDD of the top domain is 2V, and VSS of the top domain is 1V, which is the same as the VDD of the bottom domain.

With stacked domain, we can align the voltages between battery and the core and avoid power delivery inefficiency.

Furthermore, current is recycled between two domains, which saves power.

**Slide**

This slide shows our problem formulation.

Given netlist, timing constraints, level shifter timing and power models, voltage regulator efficiency and design power information, we partition the netlist into two domains, define layout region of each domain and place instances and level shifter. The objective here is to maximize battery lifetime of the design.

However, the stacked-domain optimization is not trivial. There are several challenges.

First, we have to ensure current balancing across multiple operating scenarios, such as function mode and sleep mode.

Second, region generation for power domain will introduce layout constraints. We must minimize the corresponding overheads.

Third, level shifter insertion will also incur power, area and timing penalties. We therefore want to minimize the number of level shifters.

**Slide**

This slide shows our overall optimization flow.

To achieve an estimation of the post-placement timing and current profile, we first perform a trial placement.

We then perform flow-based partitioning with layout and timing-path awareness, and multi-scenario current balancing constraints.

We then define regions for power domains. To reduce the complexity for power delivery network design and to minimize the area cost from gap insertion along the boundary between domains, we generate one continuous region for each power domain with minimized boundary length.

Based on the defined region for each power domain, we then perform re-floorplaning and insert level shifters.

Last, we perform placement optimization and incremental timing fix to remove timing violations caused by level shifter insertion.

**Slide**

This slide shows an example of our optimization flow.

The first figure shows our flow-based partitioning solution which is layout aware. In blue are instances from the bottom domain and in red are instances from the top domain.

In the second figure, we legalize the placement, define region for each power domain and optimize the boundary in between.

In the last figure, we resize the floorplan and insert level shifters which are in yellow.

I will give details of each optimization step in the following slides.

**Slide**

This slide shows the basic idea of flow-based partitioning.

Our goal here is to perform current-balanced partitioning on the netlist with minimized number of cuts.

In the partitioning flow, we first construct a flow network based on the netlist, where each cell or cluster of cells becomes a vertex, each net becomes an edge, and the weight of the vertex is estimated based on the current of the cell or cluster of cells.

We then perform max-flow optimization, which gives the min-cut partitioning according to the max-flow min-cut theorem.

After each max-flow optimization, if the currents between two partitions are not balanced, we cluster the nodes from the smaller partition, together with a neighbor note into one super vertex. We then perform another round of the max-flow optimization which gives a different partitioning solution. Note that clustering a neighbor nodes avoid the same partitioning solution in the second max-flow optimization.

We iteratively perform max-flow optimization and clustering until currents are balanced between two partitions.

The bottom figures show one example, where a and b are source and sink, and all nodes have the same weight.

On this example, the first max-flow optimization finds the min cut a-c and a-e. But the currents are not balanced. So we cluster the smaller partition, which only has node a, and the randomly-selected neighbor e and perform the max-flow optimization again. The second max-flow optimization ends up with cuts ae-c and f-b. And the currents are balanced, so we have the final solution shown in the lower-left figure.

However, this basic flow-based partitioning is not aware of cell placement and timing paths. So we propose several extensions in the next slides.

**Slide**

To reduce the runtime and improve scalability of the flow-based partitioning, we propose a pre-clustering procedure. We perform heavy edge matching optimization, which simply clusters cells with dense connection, and use clusters instead of cells for flow-based partitioning optimization. The right figure shows in which different colors indicate different clusters.

We also extend the flow-based partitioning to comprehend current balancing in multiple operating scenarios. For which, we use weighted sum of normalized currents from all scenarios as the balancing constraint.

To be aware of timing paths. We remove V-shaped vertices after each max-flow optimization. The right figure illustrates the V-shaped vertices, which are cells along one timing-critical path but across the boundary between two domains multiple times.

Last, we enable the layout awareness by detecting and removing outliers after each max-flow optimization. Here, the outliers are instances from the large-current partition but located in the small-current region.

**Slide**

Based on the partitioning solution, we define region for each power domain. Although our partitioning optimization is layout aware, there can still be separated regions for each power domain. These separated regions will increase the design complexity for power delivery network. So we want to have only one continuous region for each power domain.

To achieve this, we perform a FM-based grid optimization to move cells.

We first uniformly divide the block area into grids.

We then find outliers, which are grid outside the largest continuous region of the same domain. For example, the yellow grids in the bottom figure. And neighbor grids, which are grids adjacent to the largest continuous region of a different domain. For example the green grids in the bottom figure.

Swapping pairs of outlier and neighbor grids will generate continuous region for each domain. But there is wirelength penalty. So we calculate the wirelength cost to swap of each pair of outlier and neighbor.

We then select the pair with the minimum cost to swap.

We iterate the swap moves until there is no outlier.

**Slide**

In a stacked-domain design, gap area must be inserted along boundary between two domains. So we want to minimize the length of boundary to minimize area penalty.

We propose a dynamic programming-based approach to optimize the boundary.

We ensure that the area of each domain remains the same after our optimization, and the moved instance area are restricted.

As discussed, the objective is to minimize the boundary length.

Our DP formulation is shown here. We first index turning points from left to right along the boundary. We then optimize segment (1, j) by selecting the minimum-length combination of the optimized segment (1, i) and simplified segment (i, j) over all possible I values, while satisfying the constraints.

The bottom figures show one example, where the red segment in the left figure is the original boundary, and blue segment in the right figure is the optimized boundary.

**Slide**

Last, we insert level shifters.

To generate space for level shifter insertion, we resize the floorplan by height of required level shifter rows. This also help to preserve the trial placement solution.

We then enumerate candidate locations between two domains for level shifter insertion and perform matching-based optimization insert level shifter with minimized wirelength.

This figure shows an example of floorplan resizing and level shifter insertion.

**Slide**

This slide shows our experimental setup. We use four designs from opencores website and a dual-core M4 industrial design for our experiments.

We use these tools for SP&R and timing, power analyses.

The bottom figure shows the power efficiency of voltage regulator. We see that **the efficiency reduces with smaller current supply**.

Slide 18

This slide shows our experimental results.

The figure shows battery lifetime of stacked domain designs normalized to those of conventional designs in both function and sleep mode. The table shows number of level shifters and current in top and bottom domains.

We see that our stacked-domain optimization leads to more than 10% and 3X battery lifetime improvement in function and sleep mode respectively.

And the currents are well balanced between top and bottom domains.

The larger battery lifetime improvement in sleep mode compared to that in function mode is due to smaller current in sleep mode.

**Slide**

We also apply the stacked-domain optimization to an industrial product design, which contains dual-core M4 MCU, modem and memories. In this optimization we also include clock tree synthesis. The right figure shows the optimized design. In red and blue are cells belong to bottom and top domains, and in white are level shifters.

Based on our optimization, we achieve 15% and 2X battery lifetime improvement over the conventional design in function and sleep modes, respectively.

We also explore the tradeoff between current balancing versus level shifter cost. The table shows three partitioning solutions with different delta current and number of level shifters. We evaluate the solutions with different regulator efficiency values. We see that when the voltage regulator efficiency is high, we want more balanced current; while smaller number of level shifters is preferred when the regulator efficiency is high.

**Slide**

Now, I will give the conclusion. In this work, we propose the first comprehensive framework for stacked domain optimization. We extend the existing flow-based partitioning with several practical improvements. We achieve more than 10% and 3X battery lifetime improvement over the conventional designs in function and sleep modes.

Our ongoing works include a predictive methodology to determine the block size, and optimization with more than two domains and/or in 3DICs.