**SLIDE 1: TITLE**

Good morning everyone. My talk is on “New Applications of Learning-Based Modeling in Nanoscale Integrated-Circuit Design”.

**SLIDE 2: OUTLINE**

My talk is structured into three parts. In Part 1, I discuss two works related to improved accuracy of electrical modeling. The first work is about prediction of skews and latencies in on-chip clock distribution networks. The second work is on a methodology to proliferate golden signoff timing.

In Part 2, I describe two works related to productivity improvement through improved design and implementation-space exploration. The first work is about area and power estimation of networks-on-chip or NoC routers to enable efficient architecture-level DSE. I describe parametric modeling and metamodeling approaches that we have used in this work. The second work is on prediction of power benefits of 3D-IC implementations relative to 2D-IC implementations.

In Part 3, I describe one work to enable auxiliary physical design optimizations through modeling of complex black-box heuristics in commercial design tools. The work is on a new optimization to minimize clock skew variation across PVT corners with accurate models of post-routing signal.

**SLIDE 3: LIST OF PUBLICATIONS**

Here is a list of all my publications that have either been accepted or are under review. The publications used in Part 1 are in blue, the ones in Part 2 are in brown and the ones in Part 3 are in violet.

**SLIDE 4: MOTIVATION: VALUE SCALING GAP**

As part of our groups work on roadmapping of ITRS system drivers, we observe that lithography has continued to deliver “available” Moore’s Law scaling of transistor density growing at 2x/node. However, the “realized density” scaling has slowed down to 1.6x/node roughly around 2009. The picture on the top-right shows this gap which we at UCSD refer to as the “design capability gap”. Designers spend area, power and performance resources on reliability, variability, etc. for nanoscale IC design technologies.

The picture below shows how resources spent on guardbands lead to lost benefits in performance, power, etc. In this work, we develop a chain of models to reduce these large guardbands by enabling incremental PD optimizations.

**SLIDE 5: MOTIVATION: HIGH COSTS, TURNAROUND TIMES**

What prevents design- and implementation-space exploration today?

This figure shows that EDA tool license costs are very high. The design cost of a SOC consumer portable chip in 2013 is $45M. Further, there are no systematic methodologies to enable designers to perform DSE or ISE. To perform DSE with EDA tools implies long runtimes and the exploration is a highly iterative process. In this work, we develop a chain of models that enable fast and accurate DSE, ISE.

**SLIDE 6: OUTLINE**

I present the first work related to improved accuracy of electrical modeling, parts of which were published in DATE-2013 and SLIP-2013.

**SLIDE 7: CHALLENGE: HIGH DIMENSIONALITY**

Why is Clock Tree Synthesis or CTS prediction hard?

Because a wide range of styles and methodologies are used to synthesize clock trees. The figure shows a clock tree cartoon where the yellow triangles are the clock buffers and the green rectangles are the sinks. Inputs to synthesize a clock tree are testcases, layout contexts, tools and their knobs.

Testcases are described in Verilog RTL. They differ as designs use heterogeneous blocks and multiple clock domains, hard macros. There can be multiple layout contexts e.g., core area, aspect ratio, clock entry points. Tool flows may be different and there are multiple settings for each tool. Often design teams do not completely understand the “field of use” of a CTS tool. They tend use tools as a black box. A testcase, layout context, tool and tool knobs are called a CTS instance. Clock trees can be evaluated using multiple metrics such as power, skew, delay or latency and wirelength.

Therefore, CTS prediction is difficult due to inherent high dimensionality.

**SLIDE 8: OUR CTS TESTCASE: EXAMPLE**

Previous works such as Tsay90 propose CTS testcases r1 to r5 with sink x, y coordinates. These have been widely used until the ISPD 2010 CTS contest benchmarks. These benchmarks use a placement blockage and inverters/buffers in the tree. However, these works do not use realistic CTS instances to predict outcomes.

This schematic shows an example of our CTS testcase. It has six sink groups K1 to K6. It uses real-world clock tree structures such as Clock-gating cells or CGCs; Clock dividers; and Glitch-free clock MUX. We also use multiple levels in the hierarchy. For example, sink groups K2 and K6 are at different levels in the clock tree hierarchy.

In addition, we can change layout contexts in our testcases, e.g., core aspect ratio, placement and routing blockage, uniform and nonuniform placement sinks and multiple clock entry points. These kinds of instances can lead to accurate models for prediction.

**SLIDE 9: MODELING PARAMETERS**

Modeling parameters are important for accurate prediction. To describe a CTS instance, we use parameters to describe the microarchitecture, for example, the number of sinks,

The floorplan context, for example, core area, core aspect ratio, clock entry point, placement and routing blockage as a percentage of the core area, tool constraints, for example, maximum skew, delay, buffer and sink transition time, maximum fanout, buffer size and wire width. We also use a parameter to measure nonuniformity in sink placement.

**SLIDE 10: MODELING FLOW**

In our flow, we use Verilog RTL testcases, synthesize them using Synopsys DesignCompiler to obtain a gate-level netlist. We use floorplan and microarchitecture parameters to generate a placed DEF file. Then, we use this DEF file, tool and the nonuniformity parameters to construct a CTS instance.

Now we use two CTS tools to synthesize clock trees from the CTS instance, and extract all CTS metrics of interest. Last, we use metamodeling techniques to derive fitted models for metrics.

Metamodeling techniques derive surrogate models from actual post-CTS data. The techniques we use are HSM, MARS, RBF and KG. Previous works demonstrate that these techniques are very accurate.

**SLIDE 11: MULTICOLLINEARITY**

The generic modeling problem is described by this equation. We estimate y hat x with a regression function of parameters x and regression coefficients beta, plus a random noise. The regression function is expressed as an offset plus the sum of regression coefficients times a function of each input parameter xi.

If input parameters are linear combinations of each other, for example, aspect ratio, buffer and sink transition time and wire width, then the matrix of input parameters is ill-conditioned. It results in large variance in the regression coefficients and the relationship between the inputs and the actual response y of x become hard to determine.

Therefore, we get a bad model which results in large differences between the predicted and the actual outcomes. This is the reason, previous works e.g., C4 report large estimation errors as D is greater than or equal to 10.

**SLIDE 12: OUR SOLUTION: HHSM**

To cure these errors, we propose hierarchical hybrid surrogate modeling, HHSM.

HHSM is a divide-and-conquer approach. We divide the parameters into two sets. One set of k parameters has low collinearity whereas the other set may have high collinearity. We use variance inflation factor or VIF to determine low and high collinearity. When VIF < 5, parameters exhibit low collinearity.

We derive hybrid surrogate modeling or HSM surrogate models described in C4 for each set and combine these models using weights determined from least-squares regression.

Formally, the model is given by this equation. W1 is the weight for the set with k parameters and w2 is the weight for the set with D-k parameters.

**SLIDE 13: HHSM ACCURACY**

This plot compares skew, delay or latency, power and wirelength estimation errors between HSM and our HHSM models with D varying from eight to 13. HHSM achieves up to 4x reduction in estimation errors compared to HSM.

As D varies from eight to 13, the HHSM estimation errors vary by less than 2%. The worst-case error is less than or equal to 13%.

**SLIDE 14: USE MODEL 1: WHICH TOOL SHOULD BE USED?**

We develop methodologies using HHSM for three use models to answer questions that physical design engineers typically ask (i) Which tool should be used? (ii) How should the tool be driven, that is, the field of use? And (iii) How wrong can the model guidance be?

To answer which tool should be used, we develop the following methodology. Determine the best tool using HHSM models for a given tuple of input parameters, and compare with actual post-CTS data. If the better tools match, then the prediction is correct.

The table quantifies accuracy of this methodology. As D grows from eight, errors increase across all metrics. When D is greater than or equal to 12, the errors saturate. The worst-case error is 6.13%.

**SLIDE 15: USE MODEL 2: HOW WRONG CAN THE GUIDANCE BE?**

Model guidance is wrong when model predicts ToolA is better than ToolB but actual data shows otherwise. When guidance is wrong, we quantify the suboptimality using this equation. Ratio of difference in CTS outcomes between tools to the outcome of the better tool, expressed as a percentage.

This table shows how often the guidance is wrong under the MODEL column and the percentage suboptimality under the SUB column. The worst-case suboptimality is less than 10%.

**SLIDE 16: SUMMARY**

In this work, we study high-dimensional CTS prediction with appropriate modeling parameters. We generate realistic testcases with real world CTS structures. We propose HHSM to cure multicollinearity and report worst-case estimation error less than 13%. We develop methodologies for practical use models.

**SLIDE 17: OUTLINE**

The second work in Part 1 is about a methodology to proliferate golden signoff timing published in DATE-2014 and partially in SLIP-2013.

**SLIDE 18: MOTIVATION: DISCREPANCY IN PATH SLACK**

In any IC design flow, timing closure is a critical signoff step. There are many commercial timing signoff tools. This plot here compares path slacks from two signoff tools T1 and T2 in the X and Y axes respectively.

The netlist, SPEF and library inputs to these tools are identical. BUT, the slacks estimated by the tools diverge from the perfect correlation line. Slack can diverge by up to 110ps. For modern processors, this means around 20% difference in performance, that is, a difference of one technology node of Moore’s Law scaling.

**SLIDE 19: CHALLENGES IN TIMING SIGNOFF**

So, what can be challenges in timing signoff for design teams?

Multiple commercial tools exist and they have high license fees.

Complexities of tools grow with each release. Tools contain millions of lines of complex black-box code; tools diverge from published documentation, and use proprietary timing engines. Therefore, the correlation problem is seemingly unbounded as the space of possible timing paths, slew times, etc. is essentially infinite.

Cost and budget constraints prevent design teams from owning licenses of multiple tools. Two usage models are possible. First, to understand if they have overdesigned or underdesigned. Second, how far their implementation is from signoff at each optimization loop.

We develop the learning-based GTX tool to correlate timing signoff between tools.

**SLIDE 20: MODELING ELEMENTS OF GTX TOOL**

Our analysis shows that path slack differs due to discrepancies in cell, wire and stage delays. Path slack is calculated from the required setup time at the capture flip-flop of the path and from stage delays; these in turn are calculated from cell and wire delays in each stage.

We model setup time of launch flip-flops. Next, we model cell delay for each pin-to-pin arch of all cell types in a design by varying input slews and loads. We model wire delay by varying R, C and input slews. We use estimates of wire and cell delay models to model stage delay. Finally, we use estimates from all these models to model path slack.

Because of the layered modeling structure, we say that our method is “deep”. We do not combine individual models in an additive manner as it can result in errors being added up.

**SLIDE 21: MODELING FLOW**

Our flow works as follows. We train models using timing data obtained from artificial testcases. We validate the models using data from artificial as well as real designs to minimize both mean-square error and range of errors. We test the models using real designs. The flow bounded by the blue dotted box is a one-time effort.

A new design taped out in the technology can use cells and/or wiring configurations that are out of scope for the current fitted models. Such “new” cells/wires can introduce divergence in timing reports. We use data from new designs to test the existing models. If the error is above a threshold, we use datapoints from the new design to refine our models. We refer to this flow as “incremental modeling” as shown by the red-dotted box.

**SLIDE 22: EXPERIMENTAL SETUP AND TESTCASES**

Our design of experiments includes two foundry technology libraries and netlist, SPEF and SDC files from post-SP&R implementation.

We use real designs, such as leon3, as well as artificial testcases. Artificial testcases allow us fine-grained control of pin-to-pin arcs, input slews and loads. We use several modern machine learning techniques including artificial neural network and random forests, and use large-sized datasets of over 100K datapoints for training and testing.

**SLIDE 23: CORRELATING TWO SIGNOFF TOOLS**

This plot shows the range of errors in the Y-axis and timing in the X-axis. Original is the difference in timing between two tools and GTX is the result of estimating one tool’s timing using timing data from the other tool.

GTX reduces slack divergence from 89ps to 22ps, that is, up to 4 times. Reduction in slack divergence is a result of reduction in stage delay divergence by 5X, cell delay divergence by 8X and wire delay divergence by 9X.

**SLIDE 24: PATH SLACK FROM TIMNING REPORTS**

Here is a snippet of timing reports between two tools T1 and T2. GTX estimates timing report of T1 using reports from T2. GTX estimates of T1’s reports are shown in red and the actual report from T1 is shown in green.

Delay divergence reduces from 39ps to 0.5ps and slack divergence reduces from 249ps to 3ps.

**SLIDE 25: CORRELATING SIGNOFF AND DESIGN TOOLS**

GTX can also correlate timing between a signoff and a design implementation tool. This plot compares difference in original and GTX estimates of a leading signoff and a leading design implementation tool. The range in errors is show in the Y-axes and timing in the X-axis.

Again, GTX reduces slack divergence by 7X, from 163ps to 23ps.

**SLIDE 26: SUMMARY AND PROPOSED RESEARCH**

Timing correlation with multiple tools can help design teams fix overdesign or underdesign. Commercial signoff tools’ reports diverge significantly. We develop GTX and predictive models to reduce timing divergence by up to 6.6X. We validate GTX across multiple design, libraries and analysis modes.

Our ongoing works are to expand GTX to use CCS timing models and develop methodologies to integrate GTX into timing closure flows.

**SLIDE 27: OUTLINE**

Moving on to Part 2, I discuss parametric and metamodeling methodologies that we have developed to estimate area and power of NoC routers. This work was published in DAC-2012 and was nominated for the best paper award.

**SLIDE 28: NOC MODELING INACCURACIES SO FAR …**

Networks-on-Chip have proved to be highly scalable interconnect fabrics for many-core processor architectures and ORION is widely used tool to estimate power and area of NoCs. The picture here is of a simple wormhole router showing the router components such buffers, crossbar and the arbiter. ORION1.0 released in 2002 and ORION2.0 released in 2009 use circuit or logic templates to model each component block of the router.

So, what is the problem with modeling based on circuit templates?

First, there can be RTL code mismatch. For example, there may be an additional pipeline register used in the RTL. Second, when the RTL is synthesized, logic transformation and technology mapping significantly change the gates. For example, a bunch of NOR gates may get replaced by AND-OR-INVERT.

Furthermore, ORION2.0 does not model control logic and the templates miss implementation such as PVT corners, layout contexts, etc. These make the ORION2.0 models inaccurate when compared with actual implementation data.

This figure plots the number of instances from netlists of XBAR synthesized router RTLs – Netmaker from Cambridge and Stanford NoC and ORION2.0 in the Y-axis and the number of ports in the X-axis. As the number of ports grows, ORION2.0’s overestimation error grows to 460% at P = 10.

**SLIDE 29: PARAMETRIC MODEL DEVELOPMENT**

Here is our modeling flow to develop parametric models. We use two router RTL generators, Netmaker and the Stanford NoC, a range of microarchitectural parameters such as number of ports, VCs and buffers and the flit width and a range implementation parameter such as the clock frequency as input. We hierarchically synthesize, place and route using multiple commercial tools - Synopsys DC and Cadence RC for synthesis and Cadence SOC Encounter for place and route. We analyze the post-synthesis netlists and develop the ORION\_NEW models.

There are two approaches to estimate power and area using the new models. The first, manual approach is used to estimate gate count of each component block. Then we use information such as cell area, leakage, pin capacitances and internal energy from the technology libraries to estimate area and leakage, switching and internal power using the gate counts.

As evident, the manual approach is quick and easy and can be used for pathfinding when technology libraries are unavailable; however, fine-grained implementation details such as routed wire length and area, and power of setup buffers, etc. are missing.

The second, least-squares regression fit approach uses post-P&R standard-cell count, area, leakage, switching and internal power to estimate gate count, area and power. The LSQR approach is accurate because it captures fine-grained implementation details, but is time-consuming as it requires generating training sets obtained by simulating the post-P&R flow several times.

**SLIDE 30: RESULTS OF PARAMETRIC MODELING: NOC POWER**

In this plot, the ORION2.0 estimation errors of power are circled. Using our methodology, the average error for both Stanford and Netmaker routers are less than 5% and maximum error is less than 38% at 45nm as well as 65nm. The worst-case error reduction is up to 6.5x.

**SLIDE 31: RESULTS OF PARAMETRIC MODELING: NOC AREA**

Similarly, in area estimation, ORION2.0 has very large errors as highlighted by the circles. Our methodology reduces the average estimation error to less than 10% and maximum errors to less than 30%. Reduction of maximum errors is valuable for designers and architects as they care about minimizing the worst-case errors. The worst-case error reduction is up to 4x.

**SLIDE 32: METAMODELING WITH POST-P&R DATA**

We develop another methodology to estimate NoC area and power by using metamodeling. Similar to the parametric modeling methodology, we obtain post-P&R area and power reports by varying architectural, implementation and operational parameters. We apply metamodeling techniques such as KG, RBF, MARS and SVM and develop area and power models.

Compared to the parametric modeling methodology, metamodeling is fast because we do not need to develop the parametric models from post-synthesis data.

**SLIDE 33: METAMODELING VALIDATION AND RESULTS**

We generate a total of 256 data points and use two sizes of training and testing sets. The first set is “sparse and restricted”. It contains only 50 data points and omits higher values of microarchitectural parameters. For example, it does not include values of buffer sizes greater than seven. The second set is sparse only and contains 64 data points that are uniformly sampled using Latin Hypercube sampling.

The plots show area and power results at 45nm and 65nm. RBF performs the best with the maximum error being around 20% at 65nm for area and at 45nm for power.

**SLIDE 34: SUMMARY AND PROPOSED RESEARCH**

We develop new modeling methodologies for NoC routers that relax the template mindset. The models capture architecture, implementation and operation-level details.

Using the proposed methodologies, we reduce worst-case estimation errors by factors of up to 6.5X as compared to ORION2.0

We released ORION3.0 software on the web in Feb-2013 and it implements our parametric and metamodeling methodologies. There have been over 380 downloads since Feb-2013 and only one bug has been reported so far in Oct-2013.

Our ongoing studies include trace-level power simulation and estimation using metamodeling and to develop a delta power modeling based on post-synthesis netlists and models of switching power and buffer internal power.

**SLIDE 35: OUTLINE**

The second work in Part 2 is about prediction of 3D-IC power benefits relative to 2D-IC implementations. This is a joint work with Qualcomm Research and has been submitted to DAC-2015.

**SLIDE 36: MOTIVATION: QUANTIFY 3D BENEFITS**

To regain performance and power benefits lost due to guardbands in 2D implementations, 3D-ICs have emerged as a promising solution. However, power estimation of 3D implementations is challenging because 3D benefit varies with netlist topologies, constraints and implementation styles. Also, there is no “golden” 3D implementation flow.

To the best of our knowledge, no tool/model exists today that can predict 3D power benefits based on netlists, constraints and 2D implementations.

**SLIDE 37: IMPROVEMENTS TO THE LATEST 3D FLOW (FROM GT)**

To develop an accurate estimation tool, we need a reliable 3D flow. We obtain the latest academic 3D flow from Georgia Tech and have improved it in several ways. These include automatic handling of multiple aspect ratios, pin placement that is aspect ratio and perimeter-aware, usage of 28nm FDSOI SRAMs and an automated flow to sweep multiple key implementation parameters.

**SLIDE 38: 3D POWER ESTIMATOR**

We develop separate models for internal, switching and leakage power components. We then combine these models to estimate total power. We use artificial neural networks to develop the models and inject sensitivity at the synthesis and P&R stages in the form on WLM and table-based cap scaling.

We estimate % delta power as it can lead to more accurate estimates of actual power. For example, 10% error on actual would imply the estimate is anywhere between 72mW and 88mW. However, 10% error on the %delta means the estimate is between 79mW and 81mW.

**SLIDE 39: MODELING PARAMETERS**

Our modeling parameters include total power and cell area from synthesis with WLM cap scaling. From 2D P&R, we use max transition, max fanout, clock period, utilization, aspect ratio, PVT corners, % of memory area, internal, switching and leakage power values, wirelength and the number of buffers and inverters.

Based on our analysis, we omit some constraints and implementation parameters because they do not correlate with delta 3D power relative to 2D.

**SLIDE 40: RESULTS OF 3DPE**

We use five classes of testcases as shown in the table here. The instance count, %buffers and %sequential cells vary widely across these testcases. The CPU and GPU testcases also use SRAM. These testcases enable us to make our models generalizable.

The plots at the bottom show that we achieve highly accurate predictions with around 0.1% of average and around 10% of max-min errors.

**SLIDE 41: NEW MODEL VALIDATION: STRESS-TESTING**

We use a novel validation method to stress-test our models. We vary the input parameters over wide ranges of values and test if the model estimations are sensible for realizable netlists. Among 434 testcases, 3DPE estimates up 39% less power in 3D relative to 2D. The bottom-left plot shows that in certain cases the benefit may be ~120%, but these netlists are not physically realizable.

**SLIDE 42: MGI: MODEL-GUIDED IMPLEMENTATION METHODOLOGIES**

We demonstrate the usefulness of 3DPE in guiding designers to achieve the maximum 3D benefit. In this experiment, the goal is to determine the WLM scaling that leads to minimum 3D power. The plot shows that the default WLM cap of 1.00pF does not deliver the minimum 3D power. A value of 0.45pF is the best scaling; our model predicts a value of 0.75pF, but the suboptimality is 0.34mW or 1.62%. 3DPE can guide designers to achieve ~5% less power.

**SLIDE 43: MGI: MODEL-GUIDED IMPLEMENTATION METHODOLOGIES**

Another use model of 3DPE is to predict % delta power saving in 3D for high-utilization implementations. High-utilization implementations provide more 3D benefits but have large runtimes. However, low-utilization implementations have small runtimes but provide small 3D benefits. The table shows that 3DPE can guide designers to choose a value of utilization, aspect ratio and clock period tuple that provides the best 3D benefits. In the actual data the range of benefits is between 1.58% and 2.88%.

**SLIDE 44: SUMMARY AND PROPOSED RESEARCH**

In this work, we have extended the latest 3D flow so as to generate training data to develop an accurate 3D power benefits estimator. We develop a novel validation method to stress-test the models and demonstrate application of 3DPE in model-guided implementations.

Our ongoing works include extending 3DPE from the block-level to full SOC-level. A key challenge is to identify the right parameters as the problem becomes high-dimensional.

**SLIDE 45: OUTLINE**

In Part 3, I present one work on model-guided incremental optimization to minimize the sum of clock skew variation across PVT corners. This is a joint work with Samsung and has been submitted to DAC-2015.

**SLIDE 46: CHALLENGE: “PING-PONG” EFFECT OF MULTI-CORNER TIMING OPTIMIZATIONS**

Modern SOCs implement features such as DVFS, etc. to achieve power, performance requirements. These features require designs to be signed off at multiple PVT corners. Fixing timing violations become challenging as fixes at one corner can lead to new violations at another corner, thereby a “ping-pong” effect. Minimizing clock skew variation is a strong knob to fix this “ping-pong” effect because only datapath fixes are not very effective when the clockpaths have large variations.

Therefore, we minimize the sum clock skew variations across PVT corners. This implicitly minimizes the overall physical implementation costs.

**SLIDE 47: EXAMPLES OF (LOCAL) OPTIMIZATION MOVES**

We perform both global and local optimizations on the clock tree. As part of our local optimization we implement three types of moves. Here is an initial subtree. In Type-I move, we size and/or displace a buffer. In Type-II move, we displace the buffer and/or size one of its child nodes. In Type-III move, we change the parent of a node, that is, reassign a driver.

**SLIDE 48: OPTIMIZATION FLOW**

The figure shows our complete optimization flow and the yellow highlighted box shows the local optimization in which we use learning-based delta latency models to guide the optimization. After each move, we construct a new tree to mimic the actual tool routing and recalculate slew and delay of upstream parent node and two-levels of downstream children nodes Interpolated delay, slew using classical methods (Elmore delay, D2M, PERI) do not match those of golden timer’s analysis. The estimated routing pattern and wire delay can have discrepancy with respect to the actual ECO solution of a commercial router.

**SLIDE 49: TESTCASES, DESIGN OF EXPERIMENTS**

We create artificial testcases that resemble clock trees in SOC blocks and vary multiple parameters to generate our training data. We develop large testcases that resemble high-speed CPU blocks and memory controllers in SOCs to test our models and our optimization flow. The bottom figures show examples of floorplans and clock trees of two testcases.

**SLIDE 50: MODEL EVALUATION**

These plots show the performance of our delta latency model at the nominal PVT corner. The parameters we use to develop the model are the delta of Elmore delay for a FLUTE RSMT, Elmore delay for a STST relative to the initial tree, the delta of the bounding box area relative to the initial tree, the ratio of bbox aspect ratio and fanout after and before a move. We achieve a R-squared value of 0.975 and the maximum of the absolute errors are ~20%.

**SLIDE 51: RESULTS OF MODEL-GUIDED LOCAL OPTIMIZATION**

These plots present the results of our model-guided local optimization. The blue dots correspond to Type-I, red dots to Type-II and green dots to Type-III moves. As compared to random moves, our model-guided optimization achieves 15ns of sum of clock skew variation reduction. Overall, we reduce the sum of clock skew variations by up to 4.5% for the CLS2v1 testcase.

**SLIDE 52: SUMMARY AND PROPOSED RESEARCH**

In this work, we develop a novel framework to minimize the sum of clock skew variations across PVT corners. We use learning-based delta latency models which guide the local optimization to achieve up to 4.5% reduction in sum of clock skew variations.

Our ongoing work includes development of models to predict a buffer location for minimum skew over a continuous range of possible buffer locations and investigation of whether a worse initial start point can enable us to achieve larger skew variation reduction across corners.

**SLIDE 53: ADDITIONAL ONGOING RESEARCH**

We are also pursuing additional research with the IT teams at Qualcomm San Diego and India on two problems. The first is about resource usage prediction for physical design jobs so as to reduce wastage, which is in the order of 40X currently. Our first goal is to predict the average memory required to run a PD job to within an accuracy goal of 10%.

The second problem is to schedule resources per activity within a project in a multi-project and multi-resource scenario. IT teams within IC design companies must constantly evaluate ways to minimize cost to purchase new resources without significantly impacting project tapeout schedules. We have developed an initial MILP-based formulation with multiple penalty functions for project makespan as well as resource usage per project. The models will provide the upper bounds on resources which are the inputs to the optimization.

Now, I conclude my talk. Thank you for your attention.