# Comprehensive Die-Level Assessment of Design Rules and Layouts

Rani S. Ghaida*, Yasmine Badr†, Mukul Gupta‡, Ning Jin*, and Puneet Gupta†
* GLOBALFOUNDRIES, Inc.
† EE Department., Univ. of California, Los Angeles
‡ QUALCOMM, Inc.
rani.ghaida@globalfoundries.com, ybadr@ucla.edu, puneet@ee.ucla.edu

**Abstract— Co-development of design rules and layout methodologies is the key to successful adoption of a technology. In this work, we propose Chip-level Design Rule Evaluator (ChipDRE), the first framework for systematic evaluation of design rules and their interaction with layouts, performance, margins and yield at the chip scale (as opposed to standard cell-level). A "good chips per wafer" metric is used to unify area, performance, variability and yield. The framework uses a generated virtual standard-cell library coupled with a mix of physical design, semi-empirical, and machine-learning-based models to estimate area and delay at the chip level. The result is a unified design-quality estimate that can be computed fast enough to allow using ChipDRE to optimize a large number of complex design rules. For instance, a study of well-to-active spacing rule reveals a non-monotone dependence of rule value to chip area (although the dependence to cell area is monotone) due to delay changes coming from well-proximity effect.**

## I. Introduction

Semiconductors have fueled wealth creation, making new applications (cost-) feasible with each successive technology generation. Keeping Moore's law alive would require rapid technology changes over the next decade and beyond. Accurate projection of the design impact of device and technology changes is key for making informed technology/design decisions, thereby, ensuring timely and cost-effective development of technology and design flows.

The evaluation of technology impact on design is traditionally inferred from the evaluation of Design Rules (DRs), which are the biggest design-relevant quality metric for a technology. Unfortunately, even after decades of existence, DR evaluation is largely unsystematic and empirical in nature; it relies on limited and small-scale experiments and manufacturing tests and much on speculations based on technologists/designers experience with previous technology generations [1]–[4]. The work in [5] presents a flow for the optimization of double-patterning design rules. The method consists of an optimization loop in which rules are modified, standard-cell layouts are generated, and printability is analyzed. Although this approach, like [3,4], may be suited for exploring rules from a pure printability perspective, it does not examine the electrical effects of rules. Moreover, because actual layout generation and printability analysis are excessively time-consuming, exploring a wide range of rules and rule combinations is impractical with these approaches.

More recently, the work of [6] offers a framework for evaluating design rules, at early stages of technology development, through fast layout-topology generation of standard-cell layouts and estimation of variability and manufacturability using first-order models. This work has two major limitations. First, the evaluation was performed at the cell-level, which may lead to false conclusions because most designs are routing-limited and, hence, *not every change in cell area results in a corresponding change in chip area*. Second, delay was not evaluated but it is well-known that *delay-change can affect chip area* due to techniques like buffering and gate sizing required to meet timing requirements.

In this work, we propose Chip-level Design Rule Evaluator (ChipDRE), the first framework for systematic evaluation of design rules and their interaction with layouts, performance, margins and yield at the chip scale. ChipDRE uses a "good chips per wafer" (GCPW) metric to unify area, performance, variability and functional yield. It uses a generated virtual standard-cell library coupled with a mix of physical design and semi-empirical models to estimate area, delay and yield at the chip level. To predict the design-rule/layout impact on delay and delay variability, ChipDRE employs a Static Timing Analysis model to estimate cell-delay and a neural network-based model to predict delay-margin dependent area penalty. Chip-level area is estimated from cell area – including the delay-margin area penalty – and a cell-area to chip-area model that is calibrated using actual Synthesis, Place and Route (SPR) data. Finally, GCPW is calculated taking into consideration a chip-level functional yield estimate. The result is a unified design-quality estimate that can be computed fast enough to allow using ChipDRE to optimize a large number of complex design rules and achieve "true" design/technology co-optimization.

We make the following contributions.

- We offer ChipDRE, the first framework for collective evaluation of design rules, layout styles, and library architectures *at the chip scale*. ChipDRE is designed to be *used for design/technology co-optimization* and supports state-of-the-art technologies including FinFETs and Local Interconnects (LI). It aims at making rule generation and optimization easier and much faster. Rather than exploring the entire search space of design rules manually or with conventional compute-expensive methods, the framework can be used to quickly eliminate poor rule choices.
- We develop a cell-delay estimator and a neural network-based model to project the impact of cell-delay change on the overall chip area.
- We propose a cell-area to chip-area model to project how cell area translates into chip area.
- We evaluate the rule impact on delay and report the evaluation in terms of GCPW unifying area, performance, variability and functional yield metrics. This comprehensive evaluation allows studying interesting trade-offs that occur at the chip level like the one between variability, performance and area.
- We perform evaluation studies of major design rules at advanced nodes (some FinFET-specific) including: gate to local-interconnect spacing, gate-to-well edge spacing and fin pitch.

The remaining paper is organized as follows. Section II gives an overview of our approach. Sections III elaborates the cell-delay estimation and the virtual standard-cell layout generation including I/O pin-access estimation and supporting FinFET and local-interconnect technologies. The cell-area to chip-area model is described in Section V, while the model to predict delay-margin dependent area penalty is described in Section IV. Section VI presents the results of a number of evaluation studies at 45nm technology node using ChipDRE. Finally, Section VII gives a brief summary of the paper and some directions for future research.

## II. Overview and Standard-Cell Layout Estimation

In this section, we give an overview of ChipDRE and briefly describe its components. We also present our approach for cell-layout estimation.

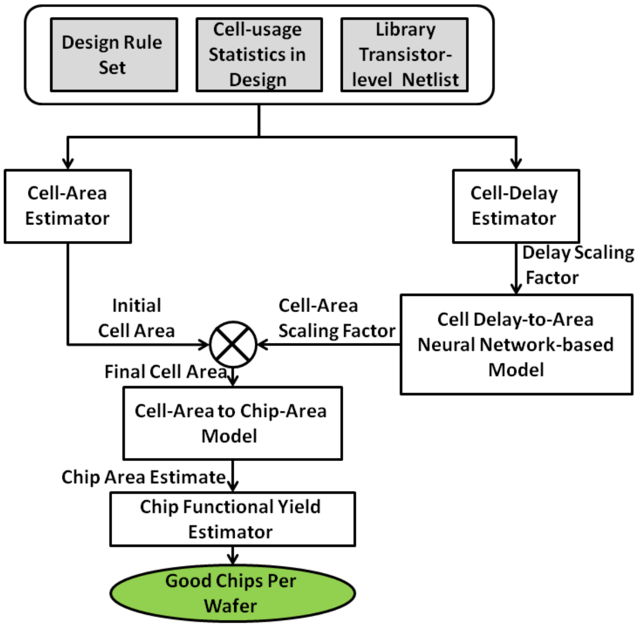

Figure 1. Overview of ChipDRE and its main components.

### A. Overview

An overview of ChipDRE is depicted in Figure 1. The framework takes the following inputs: transistor-level netlists (SPICE) of cells, rules and their values, estimates of process control (e.g., overlay error distribution), and cell-usage statistics of the design to evaluate the rules on. In ChipDRE, only the values of rules under evaluation are modified while all others remain unchanged. This modified set of rules is then used to estimate the cell-layout and perform the design-level evaluation.

Concisely, the first stage of ChipDRE is to estimate the cell layout/area and cell delay for a given set of rules. If the cell delay changes in comparison with the delay obtained using a base set of rules, the cell-delay change is converted into a delay-scaling factor which is used to scale the timing characteristics of the standard-cell library (in Liberty file format). A neural network-based model is then used to estimate the impact of cell-delay change on the design overall cell area (Figure 2 manifests the significance of this impact). The model essentially predicts gate-sizing and buffer-insertion to meet the timing requirements with the new cell-delay characteristics. In the second stage of ChipDRE, another semi-empirical model – fitted to SPR data – is used to predict how the cell area translates into chip area. The final stage of ChipDRE, chip-level functional yield is estimated and a unified design-quality metric, number of “good chips per wafer” (GCPW), is calculated.

### B. Cell-area Estimation

The cell-area estimator is based on the virtual-cell generator from [6]. This generator[1] accurately estimates cell area ($< 1\%$ error [6]) through fast generation of front-end-of-line (FEOL) layers and congestion-based estimation of wiring area. In this work, we extend the cell-layout estimator of [6] to enable its application at the chip level and using state-of-the-art technologies (e.g., FinFETs).

For chip-level evaluation, we generate I/O pin segments and the physical specifications of the technology and standard-cells (in Library Exchange Format or LEF). In studies presented in this work, pin segments are kept at minimum possible dimensions while meeting the minimum area design rule. We first sort vertical pins from left to right and horizontal pins from bottom to top. We then assign pins sequentially to the closest available track without creating DR violations. It is worth noting that we allow three pin configurations: (1) all pins on M1, (2) all pins on M2, and (3) pins on either M1 or M2 layers. In case of (3), a pin will be assigned to M1 by default and moved to M2 if doing so helps resolving M1 congestion in the cell (see Figure 3 for an example). In all our experiments, we use pin configuration (3).

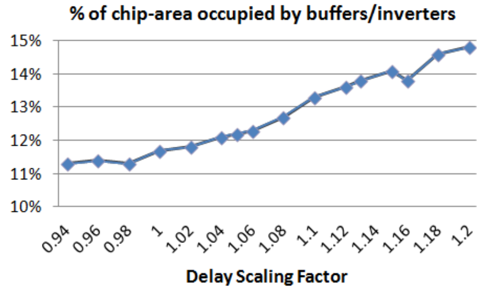

Figure 2. Empirical data from placement-aware synthesis commercial tool manifesting the impact of cell delay on the percentage of chip area that is occupied by buffers/inverters.

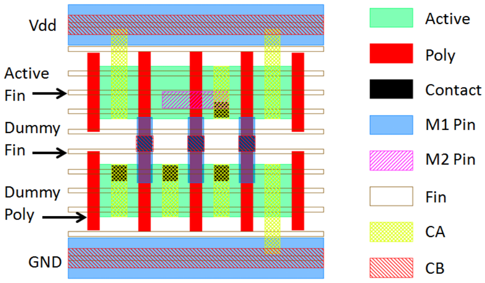

Figure 3. Example layout for OAI21_X1 cell generated by ChipDRE with FinFETs, local interconnects (i.e., CA and CB layers), and DR violation-free I/O pin segments.

FinFET technology with local-interconnect layers will be standard across the industry at advanced nodes (22nm and below [7]). Hence, to enable rule-evaluation at advanced nodes, we extend the layout-generation of front-end-of-line layers to include additional local-interconnect and FinFET-specific layers. The additional layers are: CA, CB, and fin-layer. CA is the vertical local-interconnect layer and is used to connect the fins of the same FET together[2], primarily to make contact from the contact-layer to the fins. CA can also be optionally used to make power/ground connection to the FETs (when a local-interconnect power rail exists). CB is the horizontal local-interconnect layer and is used to make contact from the contact-layer to Poly and to make short Poly-to-Poly connections when possible. The fin layer constitutes the actual FinFETs, referred to as active fins, and dummy fins, which are necessary to conform the fin layer to a grid and ensure printability. The fin grid needs to be in accordance with the cell-height so that it is maintained after cell-placement in the design. This constraint makes finding a valid configuration of fin count and pitch in active regions (P/N networks) as well as top, bottom, and center overhead regions complex. Given a range of allowed fin pitches, we run an exhaustive search to find a working configuration with maximum number of total active fins in one column and the smallest active fin pitch. To improve the chances to reach a better solution, we optionally allow the dummy fin pitch in top/bottom/center overhead regions to differ and allow the cell top/bottom edges to coincide either with the center of the fin (as in Figure 3) or with the center of the dummy fin-to-fin spacing.

To migrate a planar FET-based netlist to a FinFET-based netlist, we employ the following model to determine the number of fins for every transistor:

$$n = \lceil \frac{W}{\alpha \times F_H} \rceil, \tag{1}$$

where $W$ is the transistor width specified in the planar-based netlist, $F_H$ is the fin-height, and $\alpha$ is a planar-to-finFET width translation parameter[3].

[1] Publicly available at nanocad.ee.ucla.edu.

[2] Note this is optional when the source/drain is not contacted

[3] We use $\alpha = 2$ in our fin-pitch experiment like [8]. A higher value of $\alpha$ can be used to take into account the contribution of the top gate as well as the triangular profile of FinFETs.

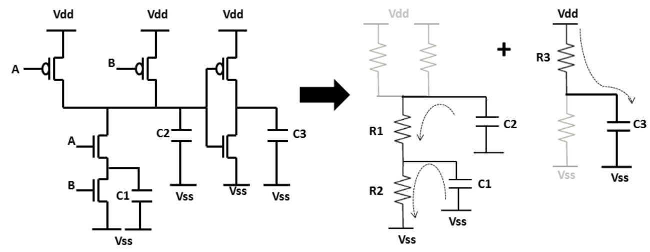

Figure 4. Estimation of low-to-high propagation delay for AND gate, equivalent RC tree and charge/discharge paths. It consists of two stages, the pull-down network for the NAND gate followed by the pull-up network of the inverter. Using Elmore delay and adding up delay stages, the propagation delay for the cell rise is estimated as: $t_{pLH}$=$R_1$ $C_1$+($R_1$+ $R_2$) $C_2$+$R_3$ $C_3$. $C_1$, $C_2$ and $C_3$ include all the gate and diffusions capacitances connected at each of the 3 nets.

The rounding up of number of fins in Equation 1 is done to ensure the minimum transistor performance is preserved after the migration.

## III. Variability-Aware Cell-Delay Estimation

A crucial aspect of Design Rule Evaluation is the assessment of the impact of the DRs on performance. To characterize a digital chip-level delay, it is required to model the delay for each standard cell. First-order delay models are employed in order to have a fast and approximate delay estimation.

### A. Cell Delay Model

To characterize the cell rise or fall delay, the cell is considered as a sequence of stages and the delays of these stages are then added up. For each stage, all paths connecting the output to the power supply (Vdd or ground) are enumerated. An RC tree is constructed for each path and Elmore delay [9] is applied to compute the path delay [10]. The worst case pull up and pull down delays are determined for each stage. Identical paths (paths that switch simultaneously) are considered as parallel resistances and their capacitances are added up.

### B. Transistor Model

We apply an RC approximation for each transistor where the capacitance model [10] considers the gate capacitance (including channel and overlap capacitances) as well as diffusion capacitances, and accounts for Miller effect. The MOS switch model in [10] is used to estimate the equivalent resistance $R_{on}$ of the transistor.

To model delay variability and consider the worst case delay, we use the current variability estimates from [6] which primarily models layout-dependent, lithography-induced variations in drive current. Variability is computed as $3\sigma$ change in current which is subtracted from the nominal current value before calculating resistance. As an example, we illustrate the pull-up of an AND gate in Figure 4.

### C. Verification and Results

For verification, we used NCX [11] with HSPICE [12] to generate the liberty file for some standard cells from Nangate Standard Cell Library [13]. The worst cases for cell rise and cell fall were compared to the values reported by ChipDRE delay estimator, using the same load capacitance.

**Gate Length Scaling Experiments.** For these experiments, the gate length rule was scaled by 10%, and the scaling factor of the ChipDRE-estimated delay (i.e. the ratio between delay at the scaled gate length to the delay at the original length) was compared to the scaling factor obtained by our spice simulation setup. Table I lists the scaling factors obtained from ChipDRE and spice, as well as the magnitude of the error which does not exceed 3%.

**Well-Proximity Effect (WPE) Experiment.** To model the Well-Proximity effect, BSIM [14] model for WPE impact on threshold voltage and mobility was used. Values of the model's parameters were computed as in [15]. The gate-to-well distance value in the BSIM model was scaled down by 10%, and the corresponding delay values were computed.

Table I
Verification of Delay model using gate-length scaling experiments by comparing the ChipDRE-estimated delay scaling factor to the scaling factor from Spice

| | Pull-up | | | Pull-down | | |
|---|---|---|---|---|---|---|
| **Cell** | **ChipDRE** | **Spice** | **Abs Error (%)** | **ChipDRE** | **Spice** | **Abs Error (%)** |
| INV_X32 | 1.10 | 1.09 | 0.9 | 1.10 | 1.07 | 3 |
| NAND2_X1 | 1.10 | 1.10 | 0.3 | 1.10 | 1.07 | 3 |
| INV_X1 | 1.08 | 1.09 | 1.1 | 1.08 | 1.07 | 1.1 |
| AND2_X4 | 1.10 | 1.09 | 0.8 | 1.10 | 1.09 | 1.2 |
| OAI21_X2 | 1.10 | 1.09 | 0.7 | 1.10 | 1.07 | 2.6 |
| AOI211_X1 | 1.10 | 1.09 | 0.5 | 1.10 | 1.08 | 2.2 |
| OAI33_X1 | 1.10 | 1.10 | 0.3 | 1.10 | 1.08 | 2.1 |
| AND2_X2 | 1.10 | 1.09 | 0.8 | 1.10 | 1.08 | 1.6 |
| Average | 1.1 | 1.09 | 0.7 | 1.1 | 1.08 | 2 |

Table II
Verification of Delay model using Well-Proximity Effect (WPE) experiment by comparing the ChipDRE-estimated delay scaling factor to the scaling factor from Spice

| | Pull-up | | | Pull-down | | |
|---|---|---|---|---|---|---|
| **Cell** | **ChipDRE** | **Spice** | **Abs Error(%)** | **ChipDRE** | **Spice** | **Abs Error(%)** |
| INV_X32 | 0.96 | 0.96 | 0.6 | 0.96 | 0.97 | 0.8 |
| NAND2_X1 | 0.76 | 0.78 | 2.4 | 0.85 | 0.88 | 2.8 |
| INV_X1 | 0.76 | 0.78 | 2.1 | 0.79 | 0.84 | 5.4 |
| AND2_X4 | 0.93 | 0.92 | 1.5 | 0.89 | 0.84 | 6.6 |
| OAI21_X2 | 0.93 | 0.93 | 0.1 | 0.93 | 0.94 | 1.0 |
| AOI211_X1 | 0.89 | 0.89 | 0.3 | 0.85 | 0.85 | 0.5 |
| OAI33_X1 | 0.89 | 0.89 | 0.8 | 0.85 | 0.88 | 3.7 |
| AND2_X2 | 0.89 | 1.00 | 11.0 | 0.87 | 0.94 | 7.3 |
| OR2_X2 | 0.87 | 0.88 | 1.4 | 0.88 | 0.88 | 0.2 |
| Average | 0.88 | 0.89 | 2.4 | 0.88 | 0.89 | 3.5 |

The ratios of cell delay with scaled gate-to-well distance to original cell delay were compared to the equivalent ratios obtained using Spice [12] simulation. Table II shows the comparison between the ratios obtained by ChipDRE to those obtained by Spice and the corresponding error which is below 7.3%.

### D. Liberty Delay File Generation

For the baseline set of design rules, we assume a Liberty file [4]. To generate the liberty file for virtual standard-cell library corresponding to the set of rules under evaluation, the worst-case pull-up and pull-down delays for the gates are computed as explained in section III-A. This is also done for the baseline set of design rules to create a reference gate delay (computed by ChipDRE). The ratios between the gate delays in the case of design rules under evaluation and those of the baseline design rules are used to scale the baseline liberty file to obtain an estimated Liberty file for the design rules under evaluation. For sequential elements, their hold and setup times are left unchanged (same as baseline liberty file), and their clock to output delay is scaled by the same scaling factor as inverter. The entire flow of generating layouts, estimating delays and generating the Liberty file within ChipDRE takes less than 49 minutes for a 100 cell library as opposed to commercial library characterization tools which take several CPU days.

## IV. Delay-to-Area Modeling

One of the major issues ChipDRE addresses which typical cell-based design rule optimization approaches suffer from is the effect of timing optimization - during physical synthesis - on area. Physical synthesis tools use several optimization techniques to meet timing constraints at the minimum possible area, like gate sizing, buffer insertion and logic restructuring. Thus, as delay of standard cell increases, we can expect an increase in the resultant chip area. Previous work [16] has experimentally characterized the impact of timing guardband reduction on some metrics of the circuit by running synthesis, place and route for several scaled

[4] This could be a characterized or scaled version from a previous technology node. The absolute values of delays in the Liberty file are not very important for ChipDRE as we are more interested in relative delay changes with rule changes.

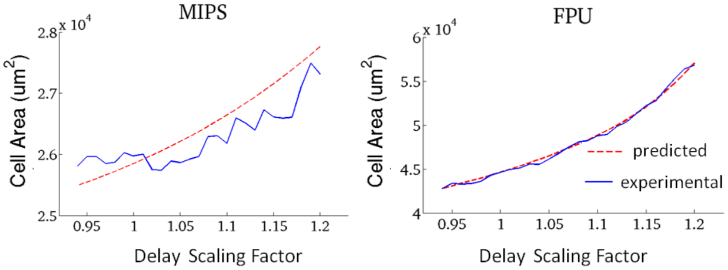

Figure 5. NN testing on MIPS design ( a blind test case) and on FPU (used in training). This network has been trained resulting in a training mean square error of 8.16x10$^4$

libraries. However, this is impractically slow to explore design rule choices. Moreover, the work of [17] has demonstrated that little noise can have huge effect on place-and-route solution quality; this makes using a model-based estimate even more attractive.

Modeling these optimization techniques analytically is complicated with a tremendous number of degrees of freedom. Thus we use a machine learning technique to predict the cell-area scaling factor (ratio between the cell area of the design at some delay scaling factor to the baseline cell area of the same design) as the standard cells delay scales (due to a change in DRs).

A neural network has been trained using data from physically-aware synthesis performed using [18][5]. To train the neural network (one hidden layer with 6 nodes), the following features have been used: number of instances on critical path, average fanout, average interconnect length, average delay and area of gates on critical path, utilization, timing constraint, ratio between area of critical paths to the total cell area and the delay scaling factor. Those features have been selected because they affect the amount of buffering and gate sizing performed by the tool to meet the timing constraints. We assumed that there is no change to the back end rules and only the front end rules are undergoing change and evaluation. Otherwise, other features need to be added like capacitance and resistance of the used metal layers per unit length.

The network was trained – using Matlab Neural Network Toolbox – on 27 delay scaling factors (each time the liberty file being scaled) from 9 test cases; 3 from [19] and 6 from ISCAS85 benchmarks. Upon testing the network on MIPS design from [19] (not used in training), the neural network was able to predict the cell-area scaling factor – used to calculate cell area – and rule out tool noise as shown in Figure 5. The figure also shows the performance of the neural network on one of the training test cases, the FPU design (from [19]).

## V. Chip Area and Yield Modeling

### A. Minimum Routable Area

Minimum Routable Area (MRA) of a design requires the estimation of maximum utilization at which the number of DR violations cease to be zero. This implies that for finding MRA multiple Place & Route (P&R) runs are required, making the whole process time consuming (detailed routing being the main culprit). For instance, an experiment to estimate MRA of AES (∼10K gate design) using binary search took 14 hours (as shown in column 6 of Table III). Such excessive runtime makes chip-level evaluation of multiple design rules impractical.

Thus, we propose a new methodology, Area Estimation using Global Routing (AEGR) that estimates MRA using global routing congestion estimates. Global routing congestion estimates require the estimation of wiring demand and wiring supply on each of the global routing cell – called G-cell – which represents a fixed number of available routing tracks in each layer. If wiring demand exceeds supply, the detailed routing is unlikely to implement a design rule correct wire pattern. Congestion in an arbitrary G-cell is given by

$$C = \frac{\text{routing demand (d)}}{\text{routing supply (s)}}. \tag{2}$$

SPR tools cannot resolve all instances of congestion and for very high congestion values, the tool might not find enough unused G-cells to successfully route the design. Hence we propose that there exists a threshold on congestion beyond which tool cannot successfully route the design. Based on this we define a metric, $m(u)$, in the following manner

$$m(u) = \alpha \times C_{peak}(u) + \beta \times C_{avg}(u), \tag{3}$$

where $C_{avg}$ is the average congestion over all G-cells and $C_{peak}$ is the maximum congestion over all G-cells , and $\alpha$ and $\beta$ are the tool dependent parameters. The utilization $u_{max}$ for which m($u_{max}$) is 1 is classified as the maximum utilization of the design.

To further refine the estimation of maximum utilization, we run detailed routing in the range $[0.9u_{max}, 1.1u_{max}]$ to get two utilization values where number of DR violations is greater than zero. Then linear extrapolation is done using these two points to estimate the utilization value where number of DR violations is equal to zero. This estimated utilization value is termed as the maximum utilization value. Using this methodology substantial runtime improvement was achieved as we show later in this section.

### B. Model Formulation

Although AEGR gives substantial improvement in runtime, it still requires running Place & Route (P&R) for all the designs and large number of FEOL design rules (increasing with every new technology node). Also, tool noise leads to problems in optimization. To overcome these problems, we model chip area as a function of total cell area thereby skipping P&R to the maximum possible extent. Our proposed model in differential form is given in Equation (4). Here $y$ is the chip area and $x$ is the total cell area. $\frac{x}{y}$ is the utilization of the design. In the proposed model, as the utilization increases or equivalently white space decreases, change in chip area is more sensitive to any change in cell area. The final analytical equation is given in Equation (5).

$$\frac{dy}{dx} = k1 - k2 \times (y/x). \tag{4}$$

After solving Equation (4), we get

$$y = \frac{k1}{k2+1} \times x + \left( y0 - (\frac{k1}{k2+1}) \times x0 \right) \times (\frac{x}{x0})^{-k2}. \tag{5}$$

There are four unknowns in the model viz. $k1$, $k2$ , $x0$ and $y0$. $y0$ can be thought of as the routing limited chip area. $x0$ can be thought of as any unutilized whitespace area[6] when the chip area is $y0$. $x0$ depends on the cell routability which in turn is dependent on the pin access and congestion within the cell [20]. Larger congestion implies router needs to drop more vias outside the cells to make connections with the cell instance pins, effectively decreasing any unutilized whitespace and hence decreasing $x0$.

To find $k1$ and $k2$ we apply the following boundary conditions

$$k1 - k2 = 1, \tag{6}$$

$$k1 - k2 \times \frac{y0}{x0} = 0. \tag{7}$$

Equation (6) is based on the fact that for very high utilization values, change in chip area is roughly equal to change in total cell area. This implies that as $u \rightarrow 1$ , $\frac{dy}{dx} \rightarrow 1$. Hence the boundary condition follows from Equation (4). Similarly from the other extreme, for any total cell area less than $x0$, chip area is routing limited and is equal to $y0$. Hence, Equation (7) follows from Equation (4). Based on these

[5] Physically-aware synthesis, which performs placement to estimate interconnect delay, has been used since it takes less time than the complete time-consuming place and route and yet produces estimates that are accurate enough for our purpose.

[6] chip area minus the area required by the router to make connection with the cell instance pins using M1 layer.

Table III
RUNTIME COMPARISON BETWEEN AREA ESTIMATION USING GLOBAL ROUTING (AEGR) METHOD AND ACTUAL P&R FOR FINDING THE MINIMUM ROUTABLE AREA.

| Design | Routing Layers | AEGR Util. | P&R Util. | Runtime in mins (AEGR) | Runtime in mins (P&R) | Runtime Reduction |
|---|---|---|---|---|---|---|
| MIPS | 3 | 0.83 | 0.83 | 97 | 322 | 3.3x |
| MIPS | 4 | 0.97 | 0.97 | 23 | 145 | 6.3x |
| JPEG | 3 | 0.93 | 0.93 | 345 | 892 | 2.6x |
| AES | 3 | 0.44 | 0.47 | 57 | 1267 | 22x |
| AES | 4 | 0.76 | 0.76 | 110 | 842 | 7.6x |
| AES | 5 | 0.85 | 0.84 | 52.4 | 141 | 2.7x |
| FPU | 3 | 0.91 | 0.90 | 52 | 261 | 5x |
| NOVA | 4 | 0.88 | 0.88 | 296 | 519 | 1.8x |

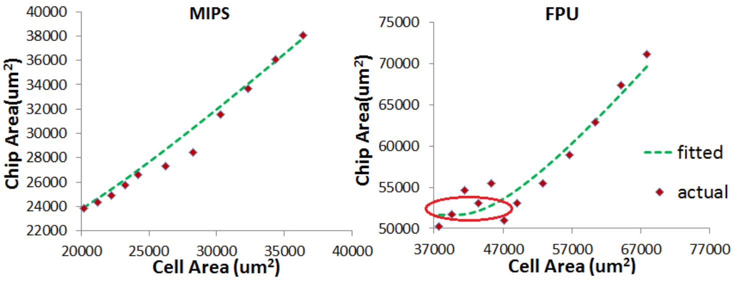

Figure 6. Plots showing MIPS and FPU chip-area vs. cell-area results obtained from actual P&R runs and those estimated using our analytical predictive model. Notice the circled region on FPU which exhibits a flat relationship between cell area and chip area. FPU is more routing-limited than MIPS.

boundary conditions, model coefficients and final analytical equation are given by

$$k1 = \frac{y0}{y0 - x0}, \quad (8)$$

$$k2 = \frac{x0}{y0 - x0}, \quad (9)$$

$$y = x + (y0 - x0) \times \left(\frac{x0}{x}\right)^{\frac{x0}{y0-x0}} \text{ for } x > x0, \quad (10)$$

$$y = y0 \text{ for } x <= x0. \quad (11)$$

Since $y0$ and $x0$ are design dependent parameters, we estimate them by actual P&R runs for each design under consideration. $x0$ and $y0$ need to be estimated only once for a given back-end interconnect stack and library architecture. This gives substantial improvement in runtime making it possible to simultaneously evaluate large number of design rules.

Our experiments to validate our methodology were performed on 5 designs from [19], synthesized using Nangate Open Cell-Library [13], and FreePDK open-source process [21]. First, data for actual P&R were created for all the designs using cadence encounter, with router objective function as "minimize congestion", and for varying number of routing layers. Based on these runs $\alpha$ and $\beta$ (in Equation (3)) were estimated to be $\frac{1}{3}$, i.e. the coefficients were estimated such that the metric agrees with the routability of designs confirmed by P&R runs. Runtime comparison between AEGR and actual P&R methods for MRA estimation is given in Table III. For actual P&R, maximum utilization was found using binary search algorithm.

To evaluate the area model, area of various cells was increased in the LEF file to closely imitate cell-area change due to FEOL design rule changes. However, the pin shapes and pin positions were not modified. Chip area was then estimated using AEGR for every increase in total cell area and the proposed model was fitted on the resulting data. The plots are shown in Figure 6 and values of $x0$ and $y0$ are shown in Table IV.

### C. Functional Yield Modeling and GCPW Calculation

Functional yield at the cell-level is computed similarly to [6]. It includes three yield-loss sources: overlay error (i.e. misalignment between layers) coupled with lithographic line-end shortening (a.k.a. pull-back), contact-hole failure, and random particle defects. The yield at the cell level is extended to the chip level using the well-known negative binomial model [7]. GCPW can then be calculated as the ratio of $\frac{wafer\_area}{chip\_area} \times yield$.

Table IV
VALUES OF $x0$ AND $y0$ FOR VARIOUS DESIGNS (SEE PLOTS OF FIGURE 6).

| Design Name | x0($um^2$) | y0($um^2$) |
|---|---|---|
| MIPS | 12526 | 20437 |
| FPU | 30950 | 36760 |

Table V
CHIP AREA COMPARISON BETWEEN GOLDEN SPR AND MODEL BASED PREDICTION ON MIPS. THE RUNTIME FOR CHIPDRE IS JUST THE CELL ESTIMATION TIME: 49 MINUTES FOR A 100 CELL LIBRARY. GOLDEN FLOW USES CHIPDRE-GENERATED LIBRARIES WITH COMMERCIAL TOOLS FOR PHYSICAL DESIGN WITH THE AEGR METHOD PROPOSED IN THIS PAPER. "EST" IS THE VALUE ESTIMATED BY CHIPDRE.

| Well-to-active spacing [nm] | Run-time (SPR) [mins] | Cell Area (est.) [$um^2$] | Chip Area (est.) [$um^2$] | Chip Area (SPR) [$um^2$] | Error in % | GCPW (est.) |
|---|---|---|---|---|---|---|
| 140 | 118 | 28171 | 30364 | 30130 | 0.8 | 667 |
| 185 | 356 | 28171 | 29709 | 29460 | 0.8 | 681 |
| 200 | 240 | 32527 | 33008 | 33913 | -2.7 | 612 |
| 210 | 207 | 32554 | 32787 | 33554 | -2.3 | 616 |

## VI. EXPERIMENTAL RESULTS

As examples, we study three interesting rules in ChipDRE: (1) well-to-active spacing rule which affects number of transistor folds (hence area and delay variability) as well as threshold voltage and mobility of transistors (hence delay); (2) local-interconnect to gate spacing rule which affects capacitances as well as area; and (3) fin-pitch rule for a candidate FinFET technology. We observe that simple cell-based estimates (as is the state-of-the-art) to assess rule quality can be misleading highlighting the importance of the ChipDRE framework [8].

### A. Well-to-active Spacing Rule Exploration

ChipDRE was used to perform a study of the well-to-active spacing rule, which impacts cell delay as well as cell area. The rule values that were chosen are 140nm, 185nm, 200nm and 210nm with 140nm as the baseline value. SPR data were generated for MIPS design using the ChipDRE-generated LEF and LIB files for each spacing rule with timing optimization done at both placement and post-routing stages while keeping the congestion effort “high”. The clock period was chosen such that minimum positive slack was achieved for the baseline case. The maximum possible cell-utilization with no DR violations and a positive timing slack is used to compute the chip area. Chip-area comparison between actual results from SPR and estimation from the proposed ChipDRE flow is given in Table V. The table also shows the GCPW metric for the design rule[9]. This study shows that a well-to-active spacing rule of value of 185nm results in the best number of GCPW even though it does not achieve the minimum cell area.

Table V results show that ChipDRE predictions are in strong agreement with the full SPR based flow and match the trends well. Interestingly, the dependence of GCPW and chip area on the rule value are *non-monotone*. This is primarily due to improved delay when well-to-active spacing is increased and despite the fact that the cell area monotonically increases as the rule value increases.

[7] Yield loss in routing-layers will be addressed in future work.

[8] We use 45nm rules from a publicly available pdk [21] to perform example studies which could be performed for future technology nodes.

[9] Note that for calculation of yield and GCPW, we assume the final design area is actually $n$ copies of the indicated area (analogous to multiple cores), where $n$ was selected to make the final design area roughly 100 $mm^2$ at the baseline design rule value.

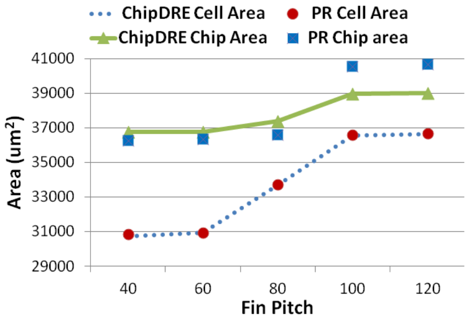

Figure 7. Plots for cell/chip area of FPU design as a function of fin pitch.

### B. FinFET Fin-Pitch Study

Fin pitch value is a technology parameter that has a strong impact on the layout density. Although fin pitch is usually defined by process and technology constraints, exploring the design implications of this rule can help process developers decide which patterning technology to adopt (e.g. Self Aligned Double Patterning vs Directed Self Assembly). We use this fin pitch exploration as an example study to highlight the difference between chip-level and cell-level assessment of DRs. Hence, we use our framework to evaluate the impact of fin pitch on cell/chip area [10]. The impact of fin pitch on delay was ignored in this experiment since its impact on parasitic capacitances was not modeled in this work. Fin pitch was varied from 60nm to 120nm in steps of 20nm and for each value standard cell layouts were generated. Based on the standard cell usage of FPU design, total cell area was computed. The cell area was then plugged into cell-area to chip-area model' and chip area was computed. This has been verified against PR runs, and the maximum error in the model predictions was found to be 5%. Figure 7 shows the chip area and cell area variations as the fin pitch is varied, both from ChipDRE and PR experiments. The figure shows that for a fin pitch of 60nm through 80nm, the cell area is steeply increasing with a very slight change in chip area, which emphasizes the importance of chip-level evaluation as opposed to cell-level evaluation. It is also observed that the fin pitch can be increased from 40nm to 60nm with a negligible impact on cell area. GCPW trends are similar to chip-area trends in this case.

### C. LI-to-gate spacing

Local interconnect is used in modern technologies to relieve congestion on local metal layers. One of the primary purposes is to make the power and ground rail connections from corresponding active areas in the devices. These connections replace contacts and metal. Unfortunately, these long contacts also increase capacitive coupling between gate and the local interconnect resulting in increased $C_{gs}$. To complicate matters further, increased spacing between gate and local interconnect can cause increase in the active area resulting in increased diffusion capacitance as well. We model both these effects in ChipDRE for the planar process and explore this spacing rule. Figure 8 shows the effect of changing the LI-to-gate spacing on the chip area (with GCPW trends being similar). In this case, the cell-area increase due to rule-value increase dominates the potential area reduction coming from delay improvement brought by a reduced gate-to-LI coupling capacitance (unlike the well-to-active rule experiment which showed a stronger delay impact).

## VII. Conclusions

We presented ChipDRE, the first framework for *fast*, *early* and *systematic* collective evaluation of design rules, layout styles, and library architectures at the chip-scale. The framework makes rule definition and optimization easier, efficient, and much more systematic. Rather than exploring the entire search space of design rules manually or with conventional compute-expensive methods, the framework can be used to quickly eliminate poor rule and technology choices. By using fast layout-estimation methods coupled with semi-empirical and neural network-based models for cell-area/cell-delay impact and trade-offs at the chip-level, the ChipDRE framework unifies area, performance, variability, and yield a "good chips per wafer" metric. To show potential applications of ChipDRE, we use it to perform evaluation studies of debatable rules for state-of-the-art technologies, including FinFETs and local-interconnects, at the chip-scale. For instance *a study of well-to-active spacing rule reveals a non-monotone dependence of rule value to chip area* (although the cell-area relationship is monotone) due to delay changes coming from well-proximity effect.

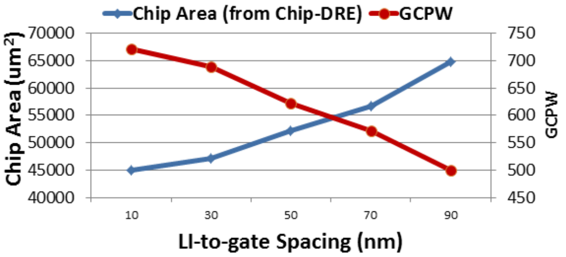

Figure 8. LI-to-gate design rule evaluation and effect on chip area for FPU.

## VIII. Acknowledgements

This work was partly supported by IMPACT+ center (http://impact.ee.ucla.edu), SRC and NSF CAREER award.

## References

[1] L. Capodieci, P. Gupta, A. B. Kahng, D. Sylvester, and J. Yang, "Toward a methodology for manufacturability-driven design rule exploration," in *Proc. DAC*, 2004, pp. 311–316.
[2] Y. Zhang, J. Cobb, A. Yang, J. Li, K. Lucas, and S. Sethi, "32nm design rule and process exploration flow," in *Proc. SPIE*, vol. 7122, 2008, p. 71223Z.
[3] V. Dai, L. Capodieci, J. Yang, and N. Rodriguez, "Developing DRC Plus rules through 2D pattern extraction and clustering techniques," in *Proc. SPIE*, vol. 7275, 2009, p. 727517.
[4] S. Chang, J. Blatchford, S. Prins, S. Jessen, T. Dam, G. Xiao, L. Pang, and B. Gleason, "Exploration of complex metal 2D design rules using inverse lithography," in *Proc. SPIE*, vol. 7275, 2009, p. 72750D.
[5] Y. Deng, Y. Ma, H. Yoshida, J. Kye, H. J. Levinson, J. Sweis, T. H. Coskun, and V. Kamat, "Dpt restricted design rules for advanced logic applications," in *Proc. SPIE*. International Society for Optics and Photonics, 2011, pp. 79 730H–79 730H.
[6] R. S. Ghaida and P. Gupta, "DRE: A Framework for Early Co-Evaluation of Design Rules, Technology Choices, and Layout Methodologies," *IEEE TCAD*, vol. 31, no. 9, Sept 2012, pp. 1379–1392.
[7] D. McGrath. Globalfoundries looks to leapfrog fab rivals. Http://www.eetimes.com/.
[8] P. Mishra, A. Muttreja, and N. K. Jha, "Finfet circuit design," in *Nanoelectronic Circuit Design*. Springer, 2011, pp. 23–54.
[9] W. C. Elmore, "The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers," *Journal of Applied Physics*, vol. 19, no. 1, 1948, pp. 55–63.
[10] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital integrated circuits-A design perspective*, 2nd ed. Prentice Hall, 2004.
[11] Synopsys Liberty NCX.
[12] (2012) Synopsys HSPICE.
[13] Nangate Open Cell Library v1.3. 2009. Http://www.si2.org/openeda.si2.org/projects/nangatelib.
[14] BSIM4.6.2 User Manual.
[15] C. M. Council. Guidelines for Extracting Well Proximity Effect Instance Parameters.
[16] K. Jeong, A. Kahng, and K. Samadi, "Impact of Guardband Reduction On Design Outcomes: A Quantitative Approach," *IEEE TSM*, vol. 22, no. 4, Nov. 2009, pp. 552 –565.
[17] A. Kahng and S. Mantik, "Measurement of inherent noise in EDA tools," in *Proc. ISQED*, 2002, pp. 206–211.
[18] Cadence RTL Compiler Advanced Physical Option.
[19] Open Cores. Http://www.opencores.org/.
[20] T. Taghavi, C. Alpert, A. Huber, Z. Li, G.-J. Nam, and S. Ramji, "New placement prediction and mitigation techniques for local routing congestion," in *Proc. ICCAD*, 2010, pp. 621–624.
[21] FreePDK. Http://www.eda.ncsu.edu/wiki/FreePDK.

[10] We realize there is no finfets in a 45nm process, but the study is performed for demonstration purposes.