New Directions for Learning-Based IC Design Tools and Methodologies

Andrew B. Kahng
UC San Diego CSE and ECE Depts., La Jolla, CA 92093 USA
abk@ucsd.edu

Abstract— Design-based *equivalent scaling* now bears much of the burden of continuing the semiconductor industry's trajectory of Moore's-Law value scaling. In the future, reductions of design effort and design schedule must comprise a substantial portion of this equivalent scaling. In this context, machine learning and deep learning in EDA tools and design flows offer enormous potential for value creation. Examples of opportunities include: improved design convergence through prediction of downstream flow outcomes; margin reduction through new analysis correlation mechanisms; and use of open platforms to develop learning-based applications. These will be the foundations of future design-based equivalent scaling in the IC industry. This paper describes several near-term challenges and opportunities, along with concrete existence proofs, for application of learning-based methods within the ecosystem of commercial EDA, IC design, and academic research.

I. Introduction: The Design Cost Crisis

It is well-understood that the *cost* of IC design has become a barrier to designers' ability to exploit advances in underlying patterning, device and integration technologies. Indeed, as far back as 2001, the *International Technology Roadmap for Semiconductors* [63] noted that "cost of design is the greatest threat to continuation of the semiconductor roadmap". The ITRS Design Cost Model [63, 40, 49] documents that non-recoverable engineering (NRE) costs are a major component of design cost; NRE includes tool license, engineer salary, compute resource and other components that typically scale with schedule. Schedule *is* scaling: the traditional Moore's Law of value scaling essentially implies that "one week equals one percent". With industry disaggregation (fabless, EDA, foundry, etc.) and consolidation (fewer companies serving as ice-breakers for the industry at the bleeding edge), a shortage of design and methodology expertise can also hamper product development in today's most advanced nodes. The recent DARPA Intelligent Design of Electronic Assets (IDEA) [55] program directly calls out today's design cost crisis, and seeks a "no human in the loop," 24-hour design framework for RTL-to-GDSII layout implementation.

Viewed more broadly, IC design faces three *intertwined* challenges: cost, quality and predictability. *Cost* corresponds to engineering effort, compute effort, and schedule. *Quality* corresponds to traditional power, performance and area (PPA) competitive metrics along with other criteria such as reliability and yield (which also determines cost). *Predictability* corresponds to the reliability of the design schedule, e.g., whether there will be unforeseen floorplan ECO iterations, whether detailed routing or timing closure flow stages will have larger than anticipated turnaround time, etc. Product QOR (quality of results) must also be predictable. The existence of unpredictability leads to guardbands and margins in the design flow that manifest as less-aggressive area utilization, clock frequency, tapeout schedule, etc. Sources of unpredictability and over-margining in the design process[1] include analysis miscorrelations (e.g., different timing slacks reported by different tools in the SP&R flow), poor estimation of downstream success vs. failure in the flow (e.g., failure of a post-synthesis netlist to ultimately close timing after placement, routing and optimization, or failure of a global routing solution to ultimately pass SI-aware timing signoff after detailed route), and inherent *tool noise* [39, 24]. Two compounding phenomena are illustrated in Figure 1 for SP&R implementation of a low-power CPU core in a foundry FinFET node enablement. First, the left side of the figure shows that variance of flow outcome (final block area, on the y-axis) increases as required design quality is increased (target frequency, on the x-axis). Samples at each target frequency are with small (~2MHz) perturbations to the target frequency in the logic synthesis stage of SP&R implementation. At the highest target frequency, nearly 11% area variation can be seen. Second, the right side of the figure shows a normal distribution of "noise" in outcome (shown: outcomes of >70 SP&R runs).

[1]We leave aside the well-documented increases in manufacturing variability, model-hardware miscorrelation, design rule complexity, signoff complexity, aggressiveness of node development timeline and yield ramp requirements, etc. in advanced nodes. Such trends only exacerbate guardbanding and margins in the design flow [28].

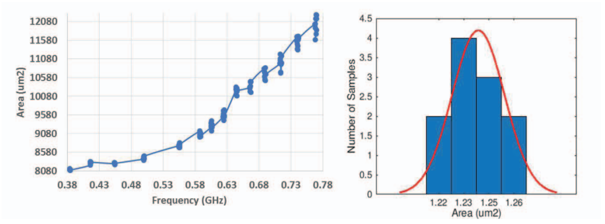


Fig. 1. Left: SP&R implementation noise increases with target design quality. Right: Observed noise is essentially Gaussian.

Design Cost and Schedule Reduction: A "Moonshot". RTL-to-GDSII turnaround time (TAT) generically spans floorplanning / powerplanning, placement, clock tree synthesis, routing and interleaved optimization steps, followed by extraction and signoff. For a single 10M-instance hard macro block in advanced nodes, a single iteration of the RTL-to-GDSII flow can easily take two weeks. Final QOR depends on innumerable methodology choices (constraint exceptions, tool commands and options used, sequencing of optimizations, etc.), with unpredictability arising from analysis complexities (e.g., dynamic voltage drop impacts on SI-aware timing) and the above-noted EDA tool noise. With unpredictability of design closure, along with the competitive requirement to minimize design guardbanding, expensive iterations are unavoidable. In this regime, advances in tool- and flow-level modeling, prediction and optimization (design-specific tuning and adaptation) have very high value. Figure I, adapted from [29], gives a pathway to game-changing reduction of design effort and cost. In the figure, the x-axis gives normalized RTL-to-GDSII schedule, and the y-axis gives normalized design QOR. (1) ML-based tool and flow predictors can enable design-adaptive flows with substantially fewer iterations (even, "one-pass"), albeit with potential QOR degradation. (2) Improved analysis correlations enable reduction of design guardbanding while retaining correct-by-construction aspects of the overall design flow. (3) Fundamental improvements in SP&R heuristics and tools, along with the leverage of cloud deployment of tools and flows, can recover overall design QOR.

Toward Learning-Based Tools and Methodologies. As reviewed in, e.g., [45, 61], machine learning has enjoyed tremendous attention and broad deployment in recent years, with highly visible successes in image classification, natural language processing, financial, robotics and biomedical applications. In *supervised learning* [42], both inputs and desired outputs are available, as in many image classification problems. In *unsupervised learning* [47], the ML application improves quality without human guidance; techniques include k-means clustering and principal component analysis, as well as recent generative adversarial network methods. Other paradigms include *semi-supervised learning*, which combines the above two methods, and *reinforcement*

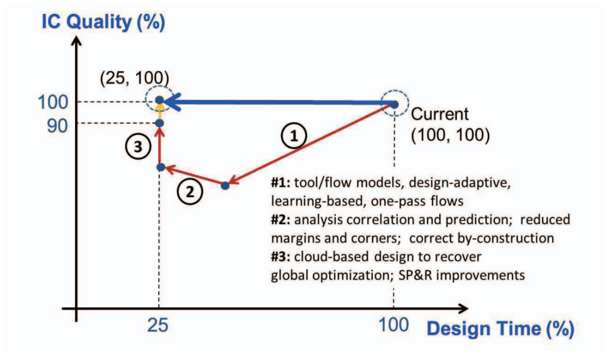


Fig. 2. "Moonshot" of design schedule and cost reduction.

learning (cf. AlphaGo [48]), which assigns rewards to actions made at particular states (e.g., of a game) based on outcomes or value achieved. At a high level, paradigms used in ML span regression, regularization, clustering, artificial neural networks (ANNs) and deep learning [23]; see [59] for one of many available taxonomies.

Previous ML applications in EDA include yield modeling for anomaly detection [14], lithography hotspot detection [17], identification of datapath regularity [50], noise and process-variation modeling [51], performance modeling for analog circuits [41], design- and implementation-space exploration [34], etc. In this paper, we broadly explore the potential for ML-based improvement of tools and flows to answer today's design cost challenge. A core premise is that realizing the vision of Figure I will *require* ML deployment both "inside" and "around" EDA tools and flows. The anticipated benefits and impacts of ML range from improved design convergence through prediction of downstream flow outcomes and model-guided optimization; margin reduction through analysis correlation mechanisms; and efficient development of learning-based applications through use of open platforms. The remainder of this paper presents concrete opportunities and directions for machine learning, focusing on RTL-to-GDSII IC implementation. Section II discusses ML techniques for improved analysis correlation, leading to improved design QOR with reduced design resources. Section III describes several techniques for tool/flow modeling and prediction, along with resulting design effort and schedule reductions. The discussion covers prediction of unroutable locations in advanced-node detailed placements, opportunities for *model-guided optimization*, and the potential for ML models to inform project- or enterprise-level optimizations that improve resource scheduling and tool usage. Conclusions and ongoing directions are given in Section IV.

II. Reduction of Analysis Miscorrelation

Analysis miscorrelation exists when two different tools return different results for the same analysis task (parasitic extraction, static timing analysis (STA), etc.) even as they apply the same "laws of physics" to the same input data. Miscorrelation forces introduction of design guardbands and/or pessimism into the flow. For example, if the P&R tool's STA report determines that an endpoint has positive worst setup slack, while the signoff STA tool determines that the same endpoint has negative worst slack, an iteration (ECO fixing step) will be required. On the other hand, if the P&R tool applies pessimism to guardband its miscorrelation to the signoff tool, this will cause unneeded sizing, shielding or VT-swapping operations that cost area, power and design schedule. Miscorrelation of timing analyses is particularly harmful: (i) timing closure can consume up to 60% of design time [19], and (ii) added guardbands not only worsen power-speed-area tradeoffs [3, 12, 19], but can also lead to non-convergence of the design. However, signoff timing is too expensive (tool licenses, incremental analysis speed, query speed, etc.) to be used within tight optimization loops.

Example: Signoff Timer Correlation. Several works apply ML to reduce the amount of iterations, turnaround time, overdesign, and license fees needed to achieve final timing signoff. The work of [33] uses a learning-based approach to fit analytical models of wire slew and delay to estimates from a signoff STA tool. These models improve the accuracy of delay and slew estimations along with overall timer correlation, such that fewer invocations of a signoff STA tool are needed during incremental gate sizing optimization [44]. The work of [22] applies deep learning to model and correct divergence between different STA tools with respect to flip-flop setup time, cell arc delay, wire delay, stage delay, and path slack at timing endpoints. Figure 3 (lower left), from [22], shows how two leading commercial STA tools T1 and T2 can differ in reported setup slack by over 100ps in a 28nm foundry technology. (Larger discrepancies are common between P&R and signoff STA analyses.) These discrepancies correspond to multiple stage delays: up to 20% of maximum performance might be lost, which is roughly equivalent to the speed benefit of a new technology node today.

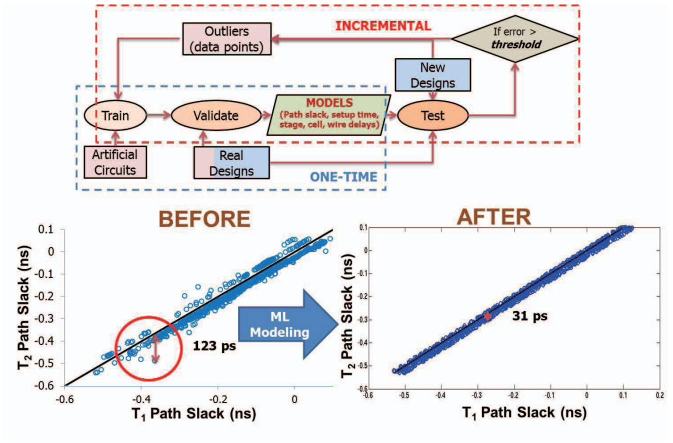


Fig. 3. Flow and results for machine learning of STA tool miscorrelation [22].

[22] learns miscorrelations between two STA tools by hierarchically composing models for discrepancies in path slack, setup delay, cell delay, wire delay and stage delay. Both linear and nonlinear ML techniques (least-squares regression (LSQR), artificial neural networks (ANN), support vector machines regression (SVMR) with radial basis function (RBF) kernel, and random forests (RF)) are applied within the modeling flow shown in Figure 3 (top). Reductions of worst-case miscorrelation by factors of 4× or more (Figure 3 (lower right)) are reported across multiple designs and foundry technologies, both with and without signal integrity (SI) analysis.

Today, the compute and license expense of signoff STA (dozens of blocks, at hundreds of mode-corner combinations, with SI and path-based analysis (PBA) options enabled) can itself be a barrier to efficient design in advanced nodes. Motivated by this, [35] achieves accurate (sub-10ps worst-case error in a foundry 28nm FDSOI technology) prediction of SI-mode timing slacks based on "cheaper, faster" non-SI mode reports. The authors of [35] identify 12 electrical and structural parameters to model the incremental transition times and arc/path delays due to SI effects.[2] Nonlinear modeling techniques, including ANNs and SVM with radial basis function (RBF) kernel, are used. Hybrid Surrogate Modeling (HSM) [36] is then used to combine predicted values from the ANN and SVM models into final predictions. Two interesting extensions are the subject of current investigation.

Extension 1: PBA from GBA. In advanced nodes, timing analysis pessimism is reduced by applying *path-based analysis* (PBA), rather than traditional *graph-based analysis* (GBA). In GBA, worst (resp. best) transitions (for max (resp. min) delay analyses) are propagated at each pin along a timing path, leading to conservative arrival time

[2]The 12 parameters are: (i) incremental delay in non-SI mode; (ii) transition time in non-SI mode; (iii) clock period; (iv) resistance; (v) coupling capacitance; (vi) ratio of coupling-to-total capacitance; (vii) toggle rate; (viii) number of aggressors; (ix) ratio of the stage in which the arc of the victim net appears to the total number of stages in the path; (x) logical effort of the net's driver; and (xi) (resp. (xii)) the differences in maximum (resp. minimum) arrival times of the signal at the driver's output pin for the victim and its strongest aggressor.

estimates at timing path endpoints. By contrast, PBA calculates path-specific transition and arrival times at each pin, reducing pessimism in endpoint arrival time estimates. Figure 4 shows the distribution of (PBA slack - GBA slack) differences per endpoint, as reported by a leading commercial signoff STA tool. The testcase is netcard [53] implemented in 28nm FDSOI with 85% utilization and 0.7ns clock period; slack differences are sorted from largest (43ps) to smallest (0ps). (Differences of 0ps for the remainder of netcard's 67,423 endpoints are not shown.) Since PBA requires $\sim 4\times$ the runtime of GBA in current tools, the ability to predict PBA-based endpoint slacks and PBA-driven timing closure optimization steps from GBA – potentially using approaches similar to those of [35] – is of high interest.

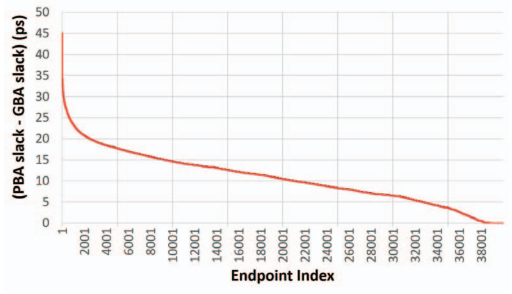
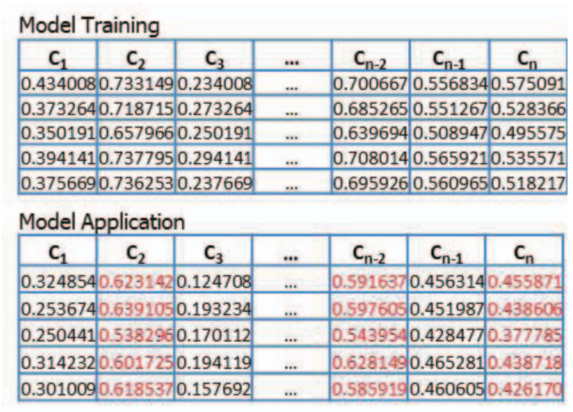


Fig. 4. Distribution of (PBA slack - GBA slack) values (ps) at endpoints of the netcard testcase implemented in 28nm FDSOI.

Extension 2: Matrix Completion. The cost of timing analysis (licenses, runtime) is roughly linear in the number of corners (PVT timing views) at which analysis is performed. In advanced nodes, signoff timing analysis is performed at 200+ corners, and even P&R and optimization steps of physical design must satisfy constraints at dozens of corners.[3] Prediction of STA results for one or more "missing" corners that *are not* analyzed, based on the STA reports for corners that *are* analyzed, would provide valuable runtime, cost and design quality. This corresponds to the well-known *matrix completion* problem in ML [5]. Figure 5 shows maximum arrival times at various design endpoints (rows of the arrays), at various analysis corners (columns of the arrays). The figure illustrates training and application of an ML model that predicts endpoint arrival times in corners C_2, C_{n-2} and C_n (red-font numbers), based on STA-determined endpoint arrival times in other (non-red) corners. The Model Training phase uses STA results at all corners to develop this ML-based prediction. However, during Model Application, timing analysis is not needed for the "missing" corners C_2, C_{n-2} and C_n. Preliminary studies suggest that single-digit picosecond accuracy of estimated endpoint slacks at "missing" corners can be achieved with surprisingly few analyzed corners; this can be very useful for internal incremental STA in timing closure and leakage optimization as in [32, 33].


Model Training

C_1	C_2	C_3	...	C_{n-2}	C_{n-1}	C_n
0.434008	0.733149	0.234008	...	0.700667	0.556834	0.575091
0.373264	0.718715	0.273264	...	0.685265	0.551267	0.528366
0.350191	0.657966	0.250191	...	0.639694	0.508947	0.495575
0.394141	0.737795	0.294141	...	0.708014	0.565921	0.535571
0.375669	0.736253	0.237669	...	0.695926	0.560965	0.518217

Model Application

C_1	C_2	C_3	...	C_{n-2}	C_{n-1}	C_n
0.324854	0.623142	0.124708	...	0.591637	0.456314	0.455871
0.253674	0.639105	0.193234	...	0.597605	0.451987	0.438606
0.250441	0.538296	0.170112	...	0.543954	0.428477	0.377785
0.314232	0.601725	0.194119	...	0.628149	0.465281	0.438718
0.301009	0.618537	0.157692	...	0.585919	0.460605	0.426170

Fig. 5. Illustration of "missing corner" analysis prediction (matrix completion).

[3]The proliferation of analysis corners is due to aggressive power management strategies (e.g., voltage scaling in logic and memories; power shut-off modes; frequency scaling) as well as manufacturing complexity (e.g., multi-patterning in BEOL layers) [28].

III. Tool/Flow Modeling and Prediction

As noted above, convergent, high-quality design requires accurate modeling and prediction of downstream flow steps and outcomes. Models (e.g., wirelength or congestion estimates) become objectives or guides for optimizations, via a "modeling stack" that reaches up to system and architecture levels. Examples include network-on-chip power and area models [56, 34], and power, area and timing models for DRAM and cache subsystems [57]. Numerous works have addressed modeling and prediction of interconnect, power, skew, wirelength, etc. [13, 9, 6] – as well as complementary structural analyses (spectral and multilevel partitioning, clustering [54]) and construction of synthetic design proxies ("eye charts") [20, 31, 58]. In this section, we give examples of tool and flow predictions that have increasing "span" across multiple design steps and that permit design feasibility assessment even at early stages of design implementation.

Example 1: Learning Model-Guided Optimization of Routability. Routability is a strong function of design rules, cell library architecture, BEOL interconnect stack [8], and previous design implementation stages. A recent sea change for physical design teams in advanced nodes is that global routing (GR) congestion no longer correlates well with design rule violations (DRCs) at the end of detailed routing. This is primarily a consequence of complex design rules that significantly constrain the detailed router. Therefore, GR-based congestion maps are no longer good predictors either for evaluating overall design routability or for identifying potential DRC violation hotspots prior to detailed routing. Figure 6 gives an example of this miscorrelation on a sub-14nm design. The figure compares a map of actual DRC violations with a map of congestion hotspots from global routing on the same layout; an overlay of these two maps is also given. Both the placer and global router are from a state-of-the-art industrial physical design platform. The netlist and layout modifications that are allowed during routing-based optimization – i.e., when DRC violations actually manifest themselves – are quite limited. Hence, an unroutable design must go through iterations back to placement in order to fix DRC violations with cell bloating, non-default routing rules, or density screens. This challenges turnaround time and motivates ML-based prediction of post-route DRC hotspots.

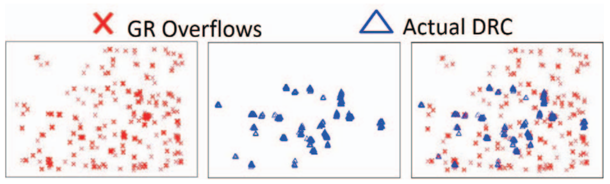


Fig. 6. Discrepancy between post-route design rule violations (DRCs) predicted from global routing overflows (left), actual post-route DRCs (middle), and an overlay of the two images (right).

Chan et al. [11] present an ML methodology and a detailed model development flow to accurately identify routing hotspots in detailed placement. The reported model incorporates parameters that are identified from deep understanding of the sub-14nm routing application; these include pin and cell densities, global routing parameters, pin proximities, multi-height cells, "unfriendly" cells whose instantiations are known to correlate with routing hotspots, etc. As shown in Figure 7, the developed ML model enables *model-guided optimization* whereby predicted routing hotspots are taken into account during physical synthesis with predictor-guided cell spreading. This helps to avoid detailed routing failures and iterations. [11] reports that for industrial benchmarks in a leading foundry sub-14nm node, the model-guided optimization reduces DRCs without timing or area overhead. Furthermore, the methodology of [11] better captures true-positives (i.e., actual routing hotspot locations) while maintaining a low false-positive rate, as shown in Figure 8. For industrial testcases, DRC count is significantly reduced by an average of 20.6% and a maximum of 76.8%, with negligible effects on timing, wirelength and design area. Going forward, convolutional neural nets (CNNs) have been successfully applied to image classification [43], and are likely well-suited to this type of P&R hotspot identification task.

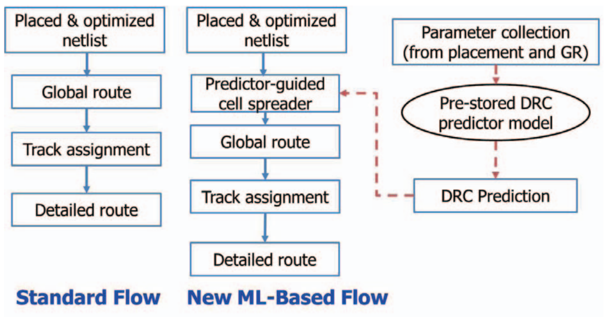
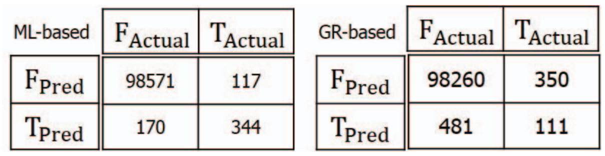

Placed & optimized netlist
Global route
Track assignment
Detailed route
Placed & optimized netlist
Predictor-guided cell spreader
Global route
Track assignment
Detailed route
Parameter collection (from placement and GR)
Pre-stored DRC predictor model
DRC Prediction
Standard Flow
New ML-Based Flow

Fig. 7. Standard P&R flow (left) vs. ML-enabled flow with model-guided placement spreading (right). Source: [11].



ML-based	F_{Actual}	T_{Actual}
F_{Pred}	98571	117
T_{Pred}	170	344

GR-based	F_{Actual}	T_{Actual}
F_{Pred}	98260	350
T_{Pred}	481	111

Fig. 8. Comparison of confusion matrices for SVM predictors, from [11]. Left: ML-based prediction has true-positive rate = 74%, false-positive rate = 0.2%. Right: Global routing-based prediction has true-positive rate = 24%, false-positive rate = 0.5%.

Example 2: ECO Route Prediction and Optimization. Complex operating scenarios in modern SOCs maximize performance while limiting power consumption. However, these techniques increase the number of mode-corner combinations at timing signoff; this in turn leads to increased datapath delay variation and clock skew variation across corners. Large timing variations increase area and power overheads, as well as design TAT, due to a "ping-pong" effect whereby fixing timing issues at one corner leads to violations at other corners. To mitigate this, [21] propose a "global-local" ML-based optimization framework to minimize the sum of clock skew variations for multi-mode multi-corner designs. Of particular interest for the present discussion is the *local optimization* in [21], which incorporates an ML-based predictor of latency changes after clock buffer and detailed routing ECOs are performed.

The local optimization iteratively minimizes skew variation via tree surgery. For a given target clock buffer, local moves include: (i) one-step sizing and displacement of the target buffer by $10\mu m$ in one of eight directions; (ii) displacement and sizing of one of the target buffer's child buffers; and (iii) reassigning the buffer to be a fanout of a different driver. Candidate local moves are tested (through placement legalization, ECO detailed routing, RC extraction and timing analysis), and the candidate move with the maximum skew variation reduction is adopted. The iterative optimization continues until there is no further improvement, or until another stopping condition is reached. Since evaluating each move involves (placement legalization, ECO routing, RC extraction, timing analysis) steps which together take minutes of runtime, it is practically infeasible to explore all move possibilities for all buffers. Instead, [21] uses machine learning to predict the driver-to-fanout latency changes induced by a given local move. For each local move, $\Delta delay$ values are calculated using low-complexity, low-accuracy analytical models. FLUTE [15] or single-trunk Steiner trees are constructed to estimate routing pattern changes, Liberty LUTs are used to estimate cell delay changes, and Elmore delay or the D2M model [2] is used to estimate wire delay changes. The new ML model, which applies Hybrid Surrogate Modeling (HSM) [36] on top of ANNs and SVM with radial basis function (RBF) kernel, then predicts actual post-ECO route $\Delta delay$ based on all the above inputs. Training data is generated using artificial testcases with various fanouts, net bounding box area and aspect ratios.

Figure 9 compares prediction accuracy of the learning-based model and analytical models. The learning-based model can identify the best move for more buffers, using fewer attempts. For real design blocks, the overall global-local skew optimization reduces the sum of skew variation over all sink pairs by up to 22%, with negligible area and power overhead.

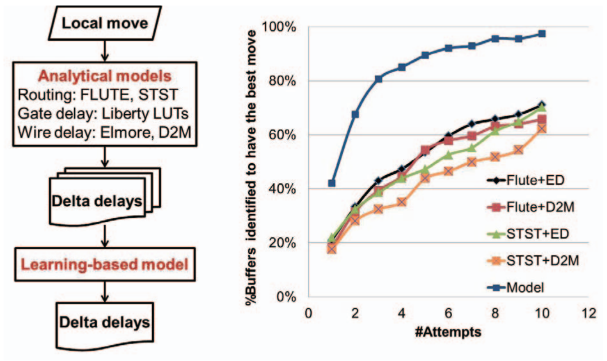


Fig. 9. Modeling flow (left) and accuracy of model prediction (right) from [21]. The ML-based model identifies the best local move for a buffer within five attempts in 90% of the trials.

Example 3: Prediction of Timing Failure Risk at Floorplan Stage. A very different ML modeling task is addressed in [7], which models and predicts a very long, complex subflow of physical implementation. Based only on netlist, constraints and floorplan information, the "multiphysics" (i.e., dynamic voltage drop-aware) endpoint timing slacks of embedded memories are predicted. This prediction must simultaneously surmount two distinct challenges. First, the complexity of timing closure in modern SOCs entails multiple iterations between various analyses and design fixes. The analyses themselves perform iterative loops across "multiphysics" such as crosstalk, IR, temperature, etc. Second, embedded memories (SRAMs) complicate the SOC layout implementation while occupying significant die area [27]. The placement of embedded memories (typically, in stacks or arrays) makes floorplanning difficult and creates P&R blockages, such that timing closure becomes more costly and difficult to predict.

The authors of [7] demonstrate that endpoint timing slacks differ significantly and non-obviously across various regimes of physics-aware analysis: (i) STA with no IR analysis; (ii) STA with static IR analysis; (iii) STA with dynamic voltage drop (DVD) IR analysis; and (iv) four iterations of STA with DVD IR analysis. They apply machine learning to achieve accurate predictive modeling of post-P&R slacks *under multiphysics timing analysis* at embedded SRAM timing endpoints, given *only* a post-synthesis netlist, floorplan and constraints. Table I lists the model parameters used in [7], which are extracted from netlist, netlist sequential graph, floorplan context and constraints. It should be emphasized that once again, the model parameter list reflects deep domain expertise and experience, along with considerable parameter and hyperparameter tuning.

TABLE I
Parameters used in ML modeling of [7].

Parameter	Description	Type	Per-memory?
N1	Max delay across all timing paths at the post-synthesis stage	Netlist	Yes
N2	Area of cells in the intersection of startpoint fanout and endpoint fanin cones of max-delay incident path	Netlist	Yes
N3	Number of stages in the max-delay incident path	Netlist	Yes
N4, N5, N6	Max, min and average product of #transitive fanin and #transitive fanout endpoints	Netlist	Yes
N7	Width and height of memory	Netlist	Yes
FP1	Aspect ratio of floorplan	Floorplan	No
FP2	Standard cell utilization	Floorplan	No
FP3, FP4	PDN stripe width and pitch	Floorplan	No
FP5	Size of buffer screen around memories	Floorplan	No
FP6	Area of blockage (%) relative to floorplan area	Floorplan	No
FP7, FP8	Lower-left placement coordinates of memories	Floorplan	Yes
FP9, FP10	Width, height of channels for memories	Floorplan	Yes
FP11	#memory pins per channel	Floorplan	Yes
C1	Sum of width and spacing of top-three routing layers after applying non-default rules (NDRs)	Constraint	No
C2	% cells that are LVT	Constraint	No
C3, C4	Max fanout of any instance in data and clock paths	Constraint	No
C5, C6	Max transition time of any instance in data and clock paths	Constraint	No
C7	Delay of the largest buffer expressed as FO4 delay	Constraint	No
C8	Clock period used for P&R expressed as FO4 delay	Constraint	No
C9	Ratio of clock periods used during synthesis and P&R	Constraint	No

We observe that [7] is another work that exploits multiple ML modeling techniques, i.e., ANN, Boosting with SVM as a weak learner, LASSO with L1 regularization and nonlinear SVM with RBF kernel. Predictions from these models are combined using HSM weights following [35]. A number of other details are specified in [7], e.g., when actual negative slack values are predicted as positive for a data point, the model is retrained by increasing the weight for the data point by $5\times$. Ultimately, experimental results in [7] show that the only 3% of the data points are false negatives (where the model suggests that a floorplan should be changed when this is actually not required). Only 4% of data points are false positives (where the model deems a floorplan to be good when it actually leads to at least one endpoint timing failure in an embedded SRAM). For data points with positive slack, the recall is 95% for 93% precision, and for data points with negative slack, the recall is 90% for 92.5% precision. These results are with respect to ground truth that is obtained from P&R and multiphysics STA reports.

Extension 1: Prediction of Doomed Runs as Time Series. Models of tool and flow outcomes can save substantial effort and schedule if they enable a "doomed run" to be avoided. (The previous example affords such savings by flagging doomed floorplans very early in the implementation flow.) Here, we briefly note that analysis of tool logfiles enables metrics to be tracked and projected as *time series*. Figure 10 shows four example progressions of the number of design rule violations during the (default) 20 iterations of a commercial router. Ideally, unsuccessful runs – i.e., those that end up with too many violations for manual fixing – should be identified and terminated after as few iterations as possible. To this end, hidden Markov models [46] can be estimated from previous examples of successful and unsuccessful runs. Or, the policy for early termination of "doomed runs" can be cast as a Markov decision process [4].

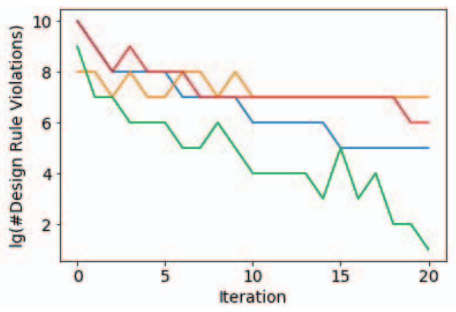


Fig. 10. Four example progressions of the number of design rule violations (shown as a base-2 logarithm) with iterations of a commercial detailed router.

Extension 2: Connection to Higher-Level Optimizations. Improved predictive models lead to improved optimization objectives, in a spirit similar to that of "model-guided optimization" above. ML-based models of tools and flows (e.g., to predict runtime and QOR outcomes) can inform planning and resource management at project and enterprise levels. This complements such works as [1], which proposes mixed integer-linear program formulations and solvers for optimal multi-project, multi-resource allocation for IC design management and cost reduction. The authors of [1] study (i) *schedule cost minimization* (SCM) to minimize overall project makespan subject to delay penalties, resource bounds, resource co-constraints, etc.; and (ii) *resource cost minimization* (RCM) to minimize the number of resources required across all projects. The work provides detailed modeling of IC projects (e.g., comprising activities such as placement, routing, RCX, STA and physical verification (PV)) and resources (e.g., CPU cores, memory, filers, and licenses for the P&R, RCX, STA and PV tools). Unique scheduling challenges are identified, such as "co-constraints" on the allocation of CPU cores and tool licenses. Figure 11 from [1] illustrates how the combined compute needs of three ongoing tapeouts can exceed the existing number of servers as well as the capacity of an IC company's datacenter. In such a situation, project schedules and resource allocations must be modified to enable the tapeouts to occur with minimum financial impact. Results reported in [1] show the potential for substantial cost (engineering headcounts, compute servers) and schedule (project makespan) gains over current practice (and, a web-based solver is available at [62]). Improved ML-based models will increase the value and impact of such optimizations.

IV. Conclusions

Machine learning techniques offer many low-hanging, high-value opportunities to the IC design and EDA industries. ML models will

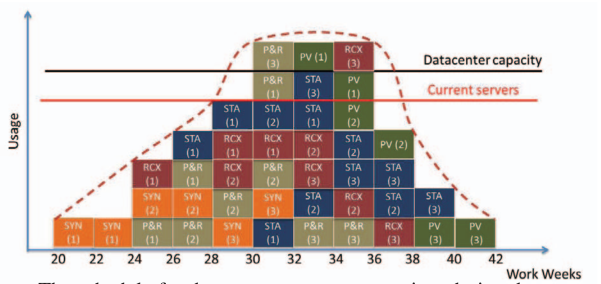


Fig. 11. The schedule for three concurrent tapeouts in a design datacenter can exceed the number of available servers, as well as the datacenter capacity.

aim to improve design convergence and QOR, e.g., via (i) pre-route estimation of SI effects to prevent unnecessary optimization of paths, and (ii) pre-route modeling of advanced technology rules to prevent unexpected QOR changes at the post-route stage of implementation. ML models will also deliver faster tool and flow TAT, e.g., via accurate estimation of signoff QOR to reduce iterations between implementation and signoff tools. And, as noted above, ML-based tool and flow predictors – likely developed initially by large IC design organizations – can even be used to guide project and resource management, e.g., for dynamic re-optimization of resource allocations to improve robustness of product development schedules. A number of potential new interactions and transfers may be spurred by ML applications within the ecosystem of commercial EDA, IC design and academic research, as shown in Figure 12.

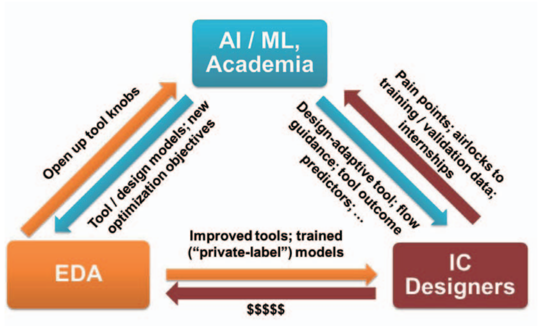

AI / ML, Academia
Open up tool knobs
Tool / design models; new optimization objectives
Design-adaptive tool; flow guidance; tool outcome predictors; ...
Pain points; airlocks to training / validation data; internships
EDA
IC Designers
Improved tools; trained ("private-label") models
$$$$$

Fig. 12. Potential ML-centered interactions within the EDA, IC design and academic research ecosystem.

The path to ML-based design cost reductions and other benefits will not be straightforward, however. While ML is often linked to the phrase "big data", ML for hardware design often takes place in a "small data" context: designs, tools, commands and options, technologies and models are always changing. Identification and extraction of modeling parameters remain an art that requires insight across circuit design, EDA algorithms, design methodology and software engineering. Further, given today's thousands of tool commands and options, multi-day runtimes, and vast space of potential designs, the scalable automations (e.g., "self-play") seen with AlphaGo and other AI/ML triumphs do not easily port to hardware design. In the IC design realm, authors of such works as [35, 11, 7] apply substantial domain expertise, along with extensive tuning of model parameters and ML model hyperparameters, to achieve their reported results. At the same time, EDA developers understand their tools, and model development effort will be naturally incented by returns on investment as well as requirements from IC designers (i.e., EDA customers).

Going forward, the challenge of "small data" may be mitigated by systematic generation of training data as in [22, 35, 21]. Furthermore, design organizations who use the same vendor tools and/or the same foundry enablements might be afforded mechanisms for data and model sharing and aggregation that inherently safeguard proprietary IP. In this light, we recall that the METRICS initiative [18, 52, 38] sought to (i) comprehensively record and measure the IC design process, (ii) develop models of EDA tools by data-mining logfiles, (iii) predict and avoid failing tool/flow outcomes, and (iv) automatically deliver the best choices of tools and runtime options for a specific design task. Standardized formats and semantics were proposed for tool logfiles, and wrapper scripts were used to automatically insert data via web interfaces into Oracle8i. Today, a platform such as Splunk [60] is commonly used to capture logfile data, along with job distribution and license usage history, at enterprise scale. And, more recent works such as [37] have shown the ability to "dial in" tool choices, command options and constraints to match required QOR metrics. Revisiting the original goals of METRICS may complement and accelerate the maturation of learning-based IC design tools and methodologies.

Finally, early investigations suggest that reinforcement learning frameworks may contribute to future "no-human-in-the-loop" reductions of design effort. For example, "multi-armed bandit" (MAB) sampling strategies can be used to achieve a resource-adaptive commercial synthesis, place and route flow that requires no human involvement. Past tool run outcomes are used to estimate the probability of meeting constraints at different parameter settings; future runs are then scheduled that are most likely to yield the best outcomes. Since the available number of runs is typically much smaller than number of tool parameter settings, domain knowledge (e.g., that meeting area/power and timing constraints becomes more difficult at higher frequency targets; recall Figure 1 above) may be incorporated into the model to improve outcomes. Preliminary studies suggest that MAB can obtain significantly better outcomes than naive sampling for a given design resource budget (licenses × schedule). Figure 13 shows the evolution of sampled frequencies versus iterations in a "no-humans" MAB implementation.

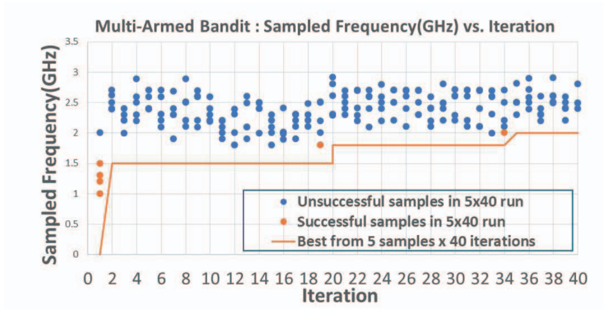


Fig. 13. Trajectory of "no-human-in-the-loop" multi-armed bandit sampling of a commercial SP&R flow, with 40 iterations and 5 concurrent samples (tool runs) per iteration. Shown: ∼2.0GHz is achieved for a small embedded CPU core in foundry FinFET technology, with user-prescribed power and area constraints. Figure adapted from [30].

V. Acknowledgments

Many thanks are due to Dr. Stefanus Mantik, Dr. Kambiz Samadi, Dr. Kwangok Jeong, Dr. Siddhartha Nath, Dr. Tuck-Boon Chan, Dr. Jiajia Li, Ms. Hyein Lee and Mr. Wei-Ting Jonas Chan who, along with other ABKGroup students and collaborators, performed much of the research reviewed in this paper. Thanks are also due to Professor Lawrence Saul for fruitful discussions and collaborations. Permission of coauthors to reproduce figures from works referenced here is gratefully acknowledged.

References

[1] P. Agrawal, M. Broxterman, B. Chatterjee, P. Cuevas, K. H. Hayashi, A. B. Kahng, P. K. Myana and S. Nath, "Optimal Scheduling and Allocation for IC Design Management and Cost Reduction", *ACM TODAES* 22(4) (2017), pp. 60:1-60:30.
[2] C. J. Alpert, A. Devgan and C. V. Kashyap, "RC Delay Metrics for Performance Optimization", *IEEE TCAD* 20(5) (2001), pp. 571-582.
[3] S. Bansal and R. Goering, "Making 20nm Design Challenges Manageable", http://www.chipdesignmag.com/pdfs/chip_design_special_DAC_issue_2012.pdf
[4] D. Bertsekas, *Dynamic Programming and Optimal Control*, Athena, 1995.
[5] E. J. Candes and B. Recht, "Exact Matrix Completion via Convex Optimization", *Foundations of Computational Mathematics* 9 (2009), pp. 717-772.
[6] Y. Cao, C. M. Hu, X. J. Huang, A. B. Kahng, S. Muddu, D. Stroobandt and D. Sylvester, "Effects of Global Interconnect Optimizations on Performance Estimation of Deep Submicron Design", *Proc. ICCAD*, 2000, pp. 56-61.
[7] W.-T. J. Chan, K. Y. Chung, A. B. Kahng, N. D. MacDonald and S. Nath, "Learning-Based Prediction of Embedded Memory Timing Failures During Initial Floorplan Design", *Proc. ASP-DAC*, 2016, pp. 178-185.
[8] W.-T. J. Chan, Y. Du, A. B. Kahng, S. Nath and K. Samadi, "BEOL Stack-Aware Routability Prediction from Placement Using Data Mining Techniques", *Proc. ICCD*, 2016, pp. 41-48.
[9] H. Chen, C. K. Cheng, A. B. Kahng, M. Mori and Q. Wang, "Optimal Planning for Mesh-Based Power Distribution", *Proc. ASP-DAC*, 2004, pp. 444-449.
[10] W.-T. J. Chan, Y. Du, A. B. Kahng, S. Nath and K. Samadi, "3D-IC Benefit Estimation and Implementation Guidance from 2D-IC Implementation", *Proc. DAC*, 2015, pp. 30:1-30:6.
[11] W.-T. J. Chan, P.-H. Ho, A. B. Kahng and P. Saxena, "Routability Optimization for Industrial Designs at Sub-14nm Process Nodes Using Machine Learning", *Proc. ISPD*, 2017, pp. 15-21.
[12] T.-B. Chan, A. B. Kahng, J. Li and S. Nath, "Optimization of Overdrive Signoff", *Proc. ASP-DAC*, 2013, pp. 344-349.
[13] T.-B. Chan, A. B. Kahng and J. Li, "NOLO: A No-Loop, Predictive Useful Skew Methodology for Improved Timing in IC Implementation", *Proc. ISQED*, 2014, pp. 504-509.
[14] T. Chen, "An ANN Approach for Modeling the Multisource Yield Learning Process with Semiconductor Manufacturing as an Example", *Computers & Industrial Engineering* 103 (2017), pp. 98-104.
[15] C. Chu and Y.-C. Wong, "FLUTE: Fast Lookup Table Based Rectilinear Steiner Minimal Tree Algorithm for VLSI Design", *IEEE TCAD* 27(1) (2008), pp. 70-83.
[16] C. Cortes and V. Vapnik, "Support-Vector Networks", *Machine Learning* 20 (1995), pp. 273-297.
[17] D. Ding, J. A. Torres and D. Z. Pan, "High Performance Lithography Hotspot Detection With Successively Refined Pattern Identifications and Machine Learning", *IEEE TCAD* 30(11) (2011), pp. 1621-1634.
[18] S. Fenstermaker, D. George, A. B. Kahng, S. Mantik and B. Thielges, "METRICS: A System Architecture for Design Process Optimization", *Proc. DAC*, 2000, pp. 705-710.
[19] R. Goering, "What's Needed to "Fix" Timing Signoff?", *DAC Panel*, 2013.
[20] P. Gupta, A. B. Kahng, A. Kasibhatla and P. Sharma, "Eyecharts: Constructive Benchmarking of Gate Sizing Heuristics", *Proc. DAC*, 2010, pp. 597-602.
[21] K. Han, A. B. Kahng, J. Lee, J. Li and S. Nath, "A Global-Local Optimization Framework for Simultaneous Multi-Mode Multi-Corner Skew Variation Reduction", *Proc. DAC*, 2015, pp. 26:1-26:6.
[22] S. S. Han, A. B. Kahng, S. Nath and A. Vydyanathan, "A Deep Learning Methodology to Proliferate Golden Signoff Timing", *Proc. DATE*, 2014, pp. 260:1-260:6.
[23] T. Hastie, R. Tibshirani and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer, 2009.
[24] K. Jeong and A. B. Kahng, "Methodology From Chaos in IC Implementation", *Proc. ISQED*, 2010, pp. 885-892.
[25] K. Jeong, A. B. Kahng, B. Lin and K. Samadi, "Accurate Machine Learning-Based On-Chip Router Modeling", *IEEE Embedded Systems Letters* 2(3) (2010), pp. 62-66.
[26] K. Jeong, A. B. Kahng and H. Yao, "On Modeling and Sensitivity of Via Count in SoC Physical Implementation", *Proc. ISOCC*, 2008, pp. 125-128.
[27] A. B. Kahng, "The Road Ahead: Product Futures", *IEEE Design & Test* 28(6) (2011), pp. 88-89.
[28] A. B. Kahng, "New Game, New Goal Posts: A Recent History of Timing Closure", *Proc. DAC*, 2015, pp. 4:1-4:6.
[29] A. B. Kahng, "PPAC Scaling at 7nm and Below", Distinguished Speaker Series talk, Cadence Design Systems, Inc., April 2016. `http://vlsicad.ucsd.edu/CDNLive2016/top.pdf`
[30] A. B. Kahng, DARPA IDEA Workshop presentation, Arlington, April 2017.
[31] A. B. Kahng and S. Kang, "Construction of Realistic Gate Sizing Benchmarks With Known Optimal Solutions", *Proc. ISPD*, 2012, pp. 153-160.
[32] A. B. Kahng, S. Kang, H. Lee, I. L. Markov and P. Thapar, "High-Performance Gate Sizing with a Signoff Timer", *Proc. ICCAD*, 2013, pp. 450-457.
[33] A. B. Kahng, S. Kang, H. Lee, S. Nath and J. Wadhwani, "Learning-Based Approximation of Interconnect Delay and Slew in Signoff Timing Tools", *Proc. SLIP*, 2013, pp. 1-8.
[34] A. B. Kahng, B. Lin and S. Nath, "ORION 3.0: A Comprehensive NoC Router Estimation Tool", *IEEE Embedded Systems Letters* 7(2) (2015), pp. 41-45.
[35] A. B. Kahng, M. Luo and S. Nath, "SI for Free: Machine Learning of Interconnect Coupling Delay and Transition Effects", *Proc. SLIP*, 2015, pp. 1-8.
[36] A. B. Kahng, B. Lin and S. Nath, "Enhanced Metamodeling Techniques for High-Dimensional IC Design Estimation Problems", *Proc. DATE*, 2013, pp. 1861-1866.
[37] A. B. Kahng, B. Lin and S. Nath, "High-Dimensional Metamodeling for Prediction of Clock Tree Synthesis Outcomes", *Proc. SLIP*, 2013, pp. 1-7.
[38] A. B. Kahng and S. Mantik, "A System for Automatic Recording and Prediction of Design Quality Metrics", *Proc. ISQED*, 2001, pp. 81-86.
[39] A. Kahng and S. Mantik, "Measurement of Inherent Noise in EDA Tools", *Proc. ISQED*, 2002, pp. 206-211.
[40] A. B. Kahng and G. Smith, "A New Design Cost Model for the 2001 ITRS", *Proc. ISQED*, 2002, pp. 190-193.
[41] T. Kiely and G. Gielen, "Performance Modeling of Analog Integrated Circuits Using Least-squares Support Vector Machines, *Proc. DATE*, 2004, pp. 448-453.
[42] S. Kotsiantis, "Supervised Learning: A Review of Classification Techniques", *Informatica*, 31 (2007), pp. 249-268.
[43] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", *Proc. NIPS* 25 (2012), pp. 1097-1105.
[44] C. W. Moon, P. Gupta, P. J. Donehue and A. B. Kahng, "Method of Designing a Digital Circuit by Correlating Different Static Timing Analyzers", *US Patent* 7,823,098, 2010.
[45] M. Mohri, A. Rostamizadeh and A. Talwalkar, "Foundations of Machine Learning", MIT press, 2012.
[46] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications on Speech Recognition", *Proc. IEEE* 77 (1989), pp. 257-286.
[47] L. K. Saul and S. T. Roweis, "Think Globally, Fit Locally: Unsupervised Learning of Low Dimensional Manifolds", *J. Machine Learning Research* 4 (2003), pp. 119-155.
[48] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. V. D. Driessche *et al.*, "Mastering the Game of Go with Deep Neural Networks and Tree Search", *Nature* 529(7587) (2016), pp. 484-489.
[49] G. Smith, "Updates of the ITRS Design Cost and Power Models", *Proc. ICCD*, 2014, pp. 161-165.
[50] S. Ward, D. Ding and D. Z. Pan, "PADE: A High-performance Placer with Automatic Datapath Extraction and Evaluation Through High Dimensional Data Learning", *Proc. DAC*, 2012, pp. 756-761.
[51] A. Zjajo, "Random Process Variation in Deep-submicron CMOS", *Stochastic Process Variation in Deep-Submicron CMOS*, pp. 17-54, Dordrecht, Springer Netherlands, 2014.
[52] The GSRC METRICS Initiative. `http://vlsicad.ucsd.edu/GSRC/metrics/`
[53] OpenCores. `http://opencores.org/projects/`
[54] Partitioning- and Placement-based Intrinsic Rent Parameter Evaluation. http://vlsicad.ucsd.edu/WLD/RentCon.pdf
[55] "DARPA Rolls Out Electronics Resurgence Initiative", `https://www.darpa.mil/news-events/2017-09-13`
[56] ORION 3.0. http://vlsicad.ucsd.edu/ORION3/
[57] CACTI7. http://vlsicad.ucsd.edu/CACTI/
[58] Gate Sizing Benchmarks With Known Optimal Solution. http://vlsicad.ucsd.edu/SIZING/bench/artificial.html
[59] Taxonomy of Machine Learning. `machinelearningmastery.com/a-tour-of-machine-learning-algorithms`
[60] Splunk. `https://www.splunk.com`
[61] Gartner Report. `https://www.gartner.com/newsroom/id/3763265`
[62] UCSD Design Cost Optimization Solver for Multi-Tapeout Project Scheduling. `http://vlsicad.ucsd.edu/MILP/`
[63] *International Technology Roadmap for Semiconductors*. `http://www.itrs2.net/itrs-reports.html`