INVITED: Reducing Time and Effort in IC Implementation: A Roadmap of Challenges and Solutions

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

ABSTRACT

To reduce time and effort in IC implementation, fundamental challenges must be solved. First, the need for (expensive) humans must be removed wherever possible. Humans are skilled at predicting downstream flow failures, evaluating key early decisions such as RTL floorplanning, and deciding tool/flow options to apply to a given design. Achieving human-quality prediction, evaluation and decision-making will require new machine learning-centric models of both tools and designs. Second, to reduce design schedule, focus must return to the long-held dream of single-pass design. Future design tools and flows that never require iteration (i.e., that never fail, but without undue conservatism) demand new paradigms and core algorithms for parallel, cloud-based design automation. Third, learning-based models of tools and flows must continually improve with additional design experiences. Therefore, the EDA and design ecosystem must develop new infrastructure for ML model development and sharing.

ACM Reference Format:
Andrew B. Kahng. 2018. INVITED: Reducing Time and Effort in IC Implementation: A Roadmap of Challenges and Solutions. In *DAC '18: The 55th Annual Design Automation Conference 2018, June 24–29, 2018, San Francisco, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3195970.3199854

1 INTRODUCTION

The semiconductor industry today faces two intertwined crises of IC design. The first crisis is *cost*: design in advanced nodes is too costly, such that designers are unable to access benefits of new technologies. This crisis was foretold as the early-1990s SEMATECH "design productivity gap" [49]; the 2001 *International Technology Roadmap for Semiconductors* (ITRS) [41] stated that "cost of design is the greatest threat to continuation of the semiconductor roadmap". The second crisis is *quality*: current design enablements and flows do not extract sufficient benefit (power, performance, area, cost, etc.) from new nodes. The 2013 ITRS roadmap highlighted a compounding *Design Capability Gap* between *available* and *realizable* transistor density benefits when designing in a new technology; see Figure 1 [17].

This paper discusses paradigm shifts that address today's cost and quality crises by reducing time and effort in IC design. The bulk of design cost is non-recoverable engineering (NRE) cost: licenses, salaries, servers, etc. These cost elements scale with headcount, tooling, and schedule (i.e., time); thus, levers for design cost reduction include the number of humans needed to perform design, the cost of design tools, and the duration of the design schedule. Because time and effort reductions can free up engineers and schedule to allow more careful design space exploration and optimization, solving the cost crisis can simultaneously help mitigate the quality crisis. The recent DARPA Intelligent Design of Electronic Assets (IDEA) program [42] [35] 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.

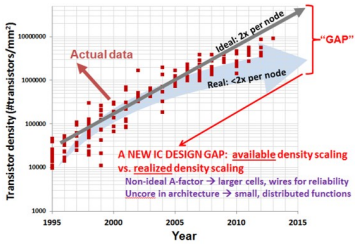

Figure 1: Design Capability Gap [41] [17].

In the following, I will discuss how the EDA and IC design communities might jointly achieve substantial time and effort reduction in IC implementation. A central theme is that machine learning (ML) techniques must pervade EDA tools, design methodologies and overall design infrastructure. Section 2 outlines basic challenges that underlie today's design cost and quality crises. Solutions are proposed, including a qualitative roadmap of ML-enabled design technology advances aimed at reducing time and effort in IC design. Section 3 discusses basic categories of ML applications in and around IC design tools, with pointers to "existence proofs" and two specific examples. Section 4 discusses industry-wide infrastructure needs, spanning standards, IP-preserving sharing mechanisms, open-source initiatives and more. The paper concludes in Section 5. Readers are referred to the contemporaneous invited papers [20] [22] for additional perspective and details.

2 CHALLENGES AND SOLUTIONS

This section gives a high-level description of challenges and solutions that lie along the path to design time and effort reductions – e.g., as targeted by the DARPA IDEA program.

Challenge 1: Breakdown of the Design Cost Model. From 2001-2014, the ITRS Design Cost Model [31, 39, 41] quantified progress of IC design technology (DT), and motivated future advances in DT, according to how well the cost of design could be controlled. *Design productivity*, expressed as the number of transistors designed per engineer-month, is central to the Design Cost Model. Together, the scaling of design productivity and the scaling of cost components (engineer salaries, server and tool license costs, etc.) enable projection of overall SOC design cost. According to the model, specific DT advances (RTL methodology, silicon virtual prototyping, electronic system-level design automation, etc.) deliver forecasted or calibrated productivity improvements when introduced. These improvements are such that *if* the DT advances are delivered on time, then design productivity will scale sufficiently to keep design costs in check.[1]

DAC '18, June 24–29, 2018, San Francisco, CA, USA

ACM ISBN 978-1-4503-5700-5/18/06…$15.00
https://doi.org/10.1145/3195970.3199854

[1] The 2013 edition of the model implies that without post-2000 DT innovations, the total design cost for the ITRS *consumer portable system-on-chip* (SOC-CP) system driver would have been at $1B in 2013, reaching $70B in 2028. Or, absent DT innovations after 2013, the total SOC-CP design cost would grow from $45.4M in 2013 to $3.4B in 2028. Thus, the Design Cost Model captures both progress and value of DT innovation.

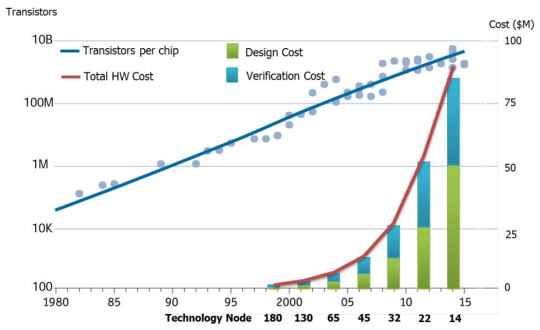

Figure 2: Design cost and transistor count trends [35].

An important observation is that the ITRS Design Cost Model had *in-built optimism*: it always envisioned some trajectory of DT innovation that would keep SOC-CP design cost under a ceiling of several tens of $M through the coming 15-year horizon. Unfortunately, (i) DT innovations are not tied lock-step to the semiconductor roadmap in the way that patterning, device and material innovations are; and (ii) they depend on creation and delivery by commercial EDA providers. Even as design tools and methodologies have advanced over the past decade, the semiconductor industry has badly diverged from the projected control of design costs, as depicted in Figure 2 [35]. The shortfall of productivity and *cost* scaling afforded by design technology has led to a cost explosion that is compounded by the above-noted shortfall (Figure 1) of *quality* scaling. The increasingly visible failing of "business as usual" motivates finding new paths forward.

Challenge 2: A Local Minimum of Design Technology and Design Quality. Both design instances and design tools become increasingly complex over time. With more heuristics deployed to meet capacity and turnaround time (TAT) requirements, tools become unpredictable, particularly when driven to their limits. Figure 3 (left), from implementation of the PULPino RISC V core in foundry 14nm enablement, shows that post-P&R area can change by 6% when target frequency changes by just 10MHz near the maximum achievable frequency. Figure 3 (right) illustrates that statistics of this noisy tool behavior are Gaussian [29] [15]. [21, 22] note that unpredictability in design implementation requires guardbanding of design targets to maintain schedule. In other words, if designers want predictable results, they must "aim low".

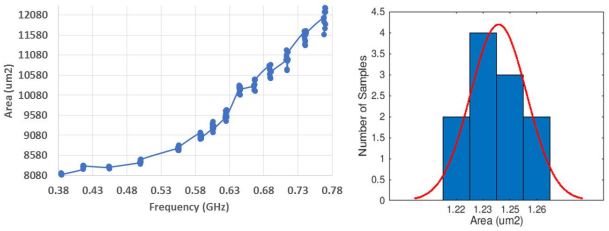

Figure 3: Left: SP&R implementation noise increases with target design quality. Right: Noise is essentially Gaussian.

Figure 4(a) suggests that with unpredictable optimizers, as well as loss of "global optimization" when problems are partitioned, designers demand near-flat methodologies. EDA developers then add more heuristics so as to process ever-larger blocks in the same TAT. To recover design quality (recall "aim low") designers seek more flexibility in their tools, to the point that a P&R tool today has well over ten thousand command-option combinations. Unfortunately, complex tools that are difficult to fathom result in unpredictable outcomes, more iterations and longer TAT, even as unpredictability also induces larger design guardbands. Consequently, *achieved* design quality worsens, and the design capability gap grows. In this way, the industry has reached a local minimum of coevolution between EDA developers' tools and IC designers' methodologies.

Solution 1: A Different DT Roadmap. A "flip the arrows" paradigm shift for future design technology R&D is suggested in Figure 4(b). The figure abandons traditional incremental improvements to QOR, capacity and TAT of an extremely complex tool, operating within an unchanging flow context.[2] Rather, the design problem is decomposed into many more small subproblems; this reduces the time needed to solve any given subproblem, and smaller subproblems can be better-solved (see [32]). To increase the number of design partitions without undue loss of global solution quality demands new placement, global routing and optimization algorithms, as well as fundamentally new RTL partition and floorplan co-optimization capabilities. Further, *reducing* design flexibility by giving designers "freedoms from choice" (RTL constructs, power distribution, etc.) can increase predictability, leading to fewer iterations and eventually single-pass design. Predictability and fewer iterations enable smaller design guardbands. The end result: better *achieved* design quality. Implied mindsets for tool developers and design flow engineers include (i) tools and flows should never return unexpected results; (ii) designers should see predictability in their tools and flows; and (iii) parallel search under the hood can preserve or improve achieved QOR. [21, 22] note that this vision strongly relies on pervasive incorporation of ML techniques.

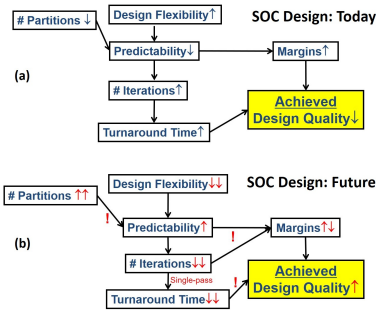

Figure 4: SOC design (a) today, and (b) in the future.

Solution 2: Machine Learning for Time and Effort Reduction. Figure 5(a) cartoons the daunting scale of design resource requirements for IC design: thousands of potential options (constraints, libraries, floorplan, envars, command options, etc.) at each flow step, along with iteration, result in an enormous tree of possible flow trajectories. Today, even identifying a "best" gate-level netlist or physical floorplan to carry forward in the flow is beyond the grasp of human engineers. Thus, the likely **first stage** of ML for time and effort reduction will entail *creating robots*: mechanizing and automating (e.g., via expert systems) 24/7 replacements for human engineers that reliably execute a given design task to completion.

Once robot engineers exist, their use must be optimized. Thus, the **second stage** of ML-based cost and effort reduction will orchestrate N robot engineers to concurrently search multiple flow trajectories; N can range from tens to thousands and is constrained chiefly by compute and license resources. Here, simple multistart, or depth-first or breadth-first traversal of the tree of flow options, is hopeless. Rather, strategies such as "go-with-the-winners" (GWTW) [2] [24], which launches multiple optimization threads, and periodically identifies and clones the most promising thread while terminating other threads (Figure 6(a)), might be applied. *Adaptive*

[2] A "big iron" analogy may apply here.

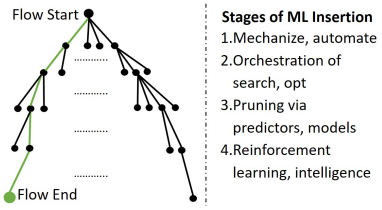

Figure 5: (a) Tree of options at flow steps. (b) Stages of ML insertion into production IC implementation.

multistart [5] [12] strategies, which exploit an inherent "big valley" structure in optimization cost landscapes to adaptively identify promising start configurations for iterative optimization, are also of interest. Figure 6(b) illustrates how better start points for optimization are identified based on the structure of (locally-minimal) solutions found from previous start points.

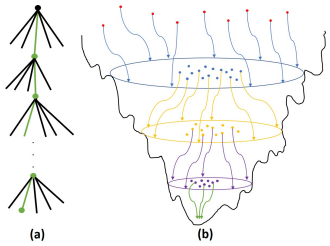

Figure 6: (a) Go-with-the-winners [2]. (b) Adaptive multi-start in a "big valley" optimization landscape [5] [12].

A **third stage** of ML insertion will integrate *prediction* of tool- and design-specific outcomes over longer and longer subflows, so as to more surgically prune, terminate, or otherwise not waste design resources on less-promising flow trajectories. Implicit in the third stage is the improvement of *predictability* and modelability for PD heuristics and EDA tools. Finally, a **fourth stage** must cover considerable remaining ground – from reinforcement learning, to "intelligence" in tools to new optimization objectives and beyond. As discussed in Section 5 below, obvious obstacles include the "small data" nature of IC design, as well as the large latencies of running today's tools and flows.

3 BASIC TYPES OF ML APPLICATIONS

This section describes three application types – along with two concrete examples – for how ML can contribute to time and effort reductions in IC design.

3.1 Toward Robot Engineers

Flow automation tools, makefile and target-based flows, load sharing facilities, and other advances have helped human engineers cope with demands of the chip design process. At the same time, Figure 2 indicates that the productivity and cost battle is being lost today. [22] observes that there are significant opportunities to address last-mile or small-market tasks that are unserved by available tools. Often, these tasks are not only time-consuming and error-prone, but also take up significant design schedule. In some cases, even expert humans can only muster trial-and-error strategies to get past a sticking point in the flow. This demands "robot engineers" that systematically search for tool command sequences, and/or observe and learn from humans. Obvious, high-value applications [22] include (i) automation of manual DRC violation fixing; (ii) automation of manual timing closure steps; (iii) placement of memory instances in a P&R block; and (iv) package layout automation.

Example: Tool Run Scheduling With a Multi-Armed Bandit. Recent work [25] shows how primitive "multi-armed bandit" (MAB) sampling might achieve resource-adaptive use of commercial synthesis, place and route with no human involvement – in a "robotic" manner distinct from expert systems approaches. In the MAB problem, we are given a slot machine with N *arms*, each arm having an unknown distribution of *rewards*. The reward obtained from each arm is i.i.d. (independent, and identically distributed); recall Figure 3. We are also given a budget of T *pulls* (also referred to as *samples*) on arms of the slot machine (more precisely, T *iterations*). The goal is to maximize the expected total reward $E[\sum_{i=1}^{T} r_{a_i}]$, where a_i is the arm played at iteration i. This involves an explore-exploit tradeoff whereby we draw samples to learn the parameters of the distributions while simultaneously maximizing rewards.[3] Studies in [25] consider several commonly-used MAB algorithms, including softmax sampling, ϵ-Greedy sampling and Thompson Sampling (TS) [38] [33] [40]. TS is found to be more robust in our design tool/flow sampling context, across a wide range of settings, compared to other algorithms. Figure 7 shows the evolution of sampled target design frequencies versus iterations using the TS algorithm. Note that the MAB sampling is inherently adaptive to its given budget of design schedule and number of tool licenses (i.e., concurrent runs). MAB scripts, code and documentation are available at [48].

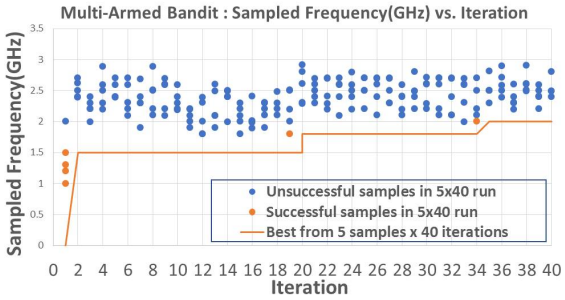

Figure 7: Trajectory of MAB sampling of a commercial SP&R flow, with 40 iterations and 5 concurrent samples (tool runs) per iteration. Testcase: PULPino in 14nm foundry technology, with given power and area constraints.

3.2 Improving Analysis Correlation

Analysis miscorrelation exists when two different tools return different results for the same input data, analysis task (parasitic extraction, static timing analysis (STA), IR drop analysis, etc.) and "laws of physics". As illustrated in Figure 8, accuracy generally comes at the cost of computation. Hence, analysis miscorrelation can be an unavoidable consequence of runtime constraints.

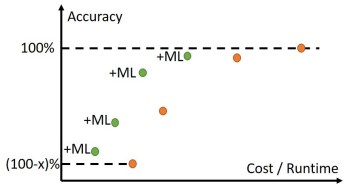

Figure 8: Accuracy-cost tradeoff in analysis.

Miscorrelation forces introduction of design guardbands and/or pessimism into the flow. If a P&R tool 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

[3] An equivalent formulation (e.g., [37]) is *regret minimization*, where *regret* is the amount lost due to not playing the optimal arm in each step, and total regret is the sum of regrets over all steps. Let r^* be the reward for the optimal arm at any step j. Then, the regret for that step is $r^* - r_{a_j}$ and the expected total regret is $E[\sum_i r^* - r_{a_i}]$.

will be required. However, if the P&R tool is overly pessimistic in guardbanding miscorrelation to signoff STA, then it will perform unneeded sizing, shielding or VT-swapping operations that cost area, power and schedule. Miscorrelation of timing analyses is particularly harmful: (i) timing closure can consume up to 60% of design time [10], and (ii) added guardbands not only worsen power-speed-area tradeoffs [3, 10], but can also lead to non-convergence.

Machine learning offers the potential to achieve "accuracy for free", shifting the cost-accuracy tradeoff curve as shown in the figure. In past work, [14] applies ML 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. [27] addresses prediction of SI-mode timing slacks. [20] suggests two near-term extensions: (1) prediction of *path-based analysis* from traditional *graph-based analysis* in STA; and (2) prediction of timing at "missing corners" that *are not* analyzed, based on STA reports for corners that *are* analyzed. ML for analysis correlation can be tightly linked to the prediction of tool and flow outcomes discussed in the next subsection. This is due to the symbiosis between design analysis and optimization. Example applications include correlation of "multiphysics" analysis flows and loops (e.g., involving temperature and voltage droop in combination with signal integrity-aware timing [7] [19]) at full-chip or die-package levels.

3.3 Predictive Modeling of Tools and Designs

A one-pass design process requires accurate modeling and prediction of downstream flow steps and outcomes, since neither loops nor excessive margins can be tolerated. Future machine learning-based predictive models (e.g., of wirelength, congestion, timing, power, etc.) have a dual purpose, in that they also serve as objectives or guides for optimizations, via a "modeling stack" that reaches up to system, architecture, and even project and enterprise levels.[4] Four supporting types of machine learning applications arise in many contexts throughout IC design: (i) identification of structural attributes of design instances that determine flow outcomes; (ii) identification of "natural structure" in designs (cf. [44]) that will permit extreme partitioning and decomposition; (iii) construction of synthetic design proxies ("eye charts") [11, 23, 45] that enable characterization of tools and flows; and (iv) prediction of the "fixed point" of a given chicken-egg loop of design (e.g., the loop between floorplanning and global interconnect design, or the loop between placement and power distribution).

Tool and flow predictions must also increase their "span" across multiple design steps: essentially, we must predict what will happen at the end of a longer and longer "rope" of design steps when the rope is wiggled. [20] reviews several works that give a progression of "longer ropes": (i) prediction from global/trial routing through detailed routing and from ECO placement through incremental global/trial routing [8]; (ii) prediction from clock buffer and topology change through automated placement and routing ECOs, extraction, and timing analysis [13]; and (iii) prediction from netlist and floorplan information through placement, routing, optimization and IR drop-aware timing analysis [7].

Example: Predicting Doomed Runs. Time and effort can be saved by predicting whether a tool run is "doomed". Modern detailed routers default to 20-40 iterations which can take many days of runtime. If detailed routing runs with an inevitably excessive number of design rule violations (DRVs) can be stopped early, then resources and schedule can be repurposed. The same applies to doomed P&R flows, doomed floorplans, etc.

Figure 9 shows four example progressions of DRVs during the (default) 20 iterations of a commercial router. We wish to identify

[4] [1] shows that project- and enterprise-level schedule and resource optimizations, supported by accurate estimates, have the potential to achieve substantial design cost reductions.

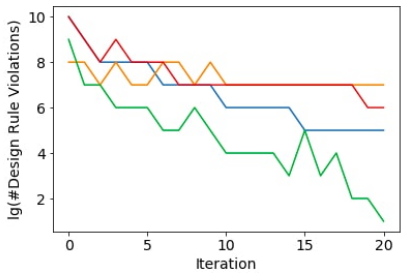

Figure 9: Example progressions of DRVs (log scale) versus iterations of a commercial detailed router.

and terminate unsuccessful runs (e.g., red and orange) that end up with too many DRVs for manual fixing, while ultimately successful runs (e.g., green) are allowed to run to completion. Tool logfile data can be viewed as *time series* to which hidden Markov models [36] or policy iteration in Markov decision processes (MDPs) [4] may be applied. For the latter, collected logfiles from previous successful and unsuccessful tool runs can serve as the basis for automated extraction of a "blackjack strategy card" for a given tool, where "hit" analogizes to continuing the tool run for another iteration, and "stay" analogizes to terminating the tool run.

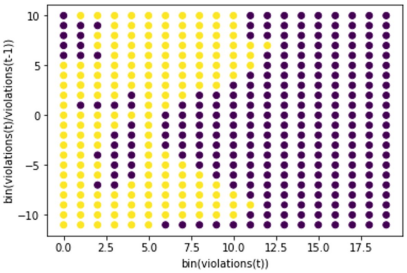

Figure 10: MDP-based "strategy card".

In [30], we have used a *state space* in the MDP that consists of binned violation count and change in DRVs since a previous iteration. *Actions* are either "GO" or "STOP", and *rewards* at each state used to derive the policy include a small negative reward for a non-stop state, a large positive reward for termination with low DRVs, etc. Figure 10 shows an MDP-based strategy card that is automatically derived from 1400 logfiles of an industry tool; each dot represents "GO" (yellow) or "STOP" (purple). The x- and y-axes represent binned violations at time t, and change in DRVs since previous iteration, respectively. We see that the MDP suggests STOP when DRVs at $t = T$ is very large (right half of the card), and suggests GO when DRVs is small. The MDP also suggests GO when DRVs is moderately large (bins 3-5) but with a negative slope.[5]

To assess the MDP-based strategy, we define two types of errors. *Type 1* errors occur when the policy stops a run that would have succeeded (where success is defined as the detailed routing run ending with ≤200 DRVs). *Type 2* errors occur when the policy allows a run to go to completion, but the run fails. We find that the policy that we produce is oversensitive, in that it stops the tool

[5] Since "training" logfiles do not generally contain information for all parts of the strategy card, we programmatically fill in the missing parts as: (i) large violations and positive slope should be STOP, (ii) small violations and large positive slope should be STOP, (iii) very large violations should be STOP, and (iv) everything else should be GO.

run too quickly. Accuracy is improved by requiring consecutive STOP signals before terminating the tool run. The table below shows training (1200 logfiles from artificial layouts) and testing (3742 logfiles from floorplans of an embedded CPU) errors when we require 1, 2 and 3 consecutive STOP signals from the policy before actually stopping the tool run. The error rate in testing is ~4% if we wait until the MDP gives three consecutive STOP signals. For the runs that are doomed, substantial iterations are saved.

Errors	Training (1200 logfiles)			Testing (3742 logfiles)		
N = 200	Total Training Error	#Type 1 Errors (wrong STOP prediction)	#Type 2 Errors (no STOP)	Total Training Error	#Type 1 Errors (wrong STOP prediction)	#Type 2 Errors (no STOP)
1 STOP	29.66%	251	99	35.3%	1317	3
2 consecutive STOPs	10.5%	27	99	8.3%	307	3
3 consecutive STOPs	8.5%	3	99	4.2%	154	3

4 INFRASTRUCTURE NEEDS

To enable the above-described paths to ML-enabled design time and effort reductions, considerable new infrastructure is required. For example, standards for ML model encapsulation, model application, IP-preserving model sharing, etc. will likely be required before any training data, data generation tasks, or ML models can be shared across multiple organizations. Specific infrastructure requirements, while too numerous to list, include the following.[6]

(1) Design owners, foundries and EDA should be comfortable that their IP (design function, technology parameters, protected syntax, etc.) is sufficiently protected (e.g., by standard anonymization and obfuscation mechanisms).

(2) ML researchers, foundries and EDA tool users should be comfortable that their use of ML to improve IC design enablement and flows does not risk IP infringement claims (e.g., would modeling a delay calculator's "error versus SPICE" be somehow prohibited?).

(3) Academic researchers should be attracted with a critical mass of ML modeling challenges, supporting data, and incentives (e.g., a "Kaggle for machine learning in IC design").

(4) In the longer term, the design, EDA and research communities must share responsibility for a "standard ML platform for EDA and IC design modeling" that spans design metrics collection, tool and flow model generation, design-adaptive tool and flow configuration, and prediction of tool and flow outcomes.

Recalling the METRICS System. The "standard ML platform for EDA and IC design modeling" requirement recalls the METRICS initiative, proposed nearly 20 years ago as a standard platform and industry infrastructure for measurement of the IC design process [9, 28, 43]. If only to help avoid repeating the past, the remainder of this section reviews METRICS and several of its lessons.

Architecture. As developed in 1999-2000, METRICS instruments design tools and design processes for continuous collection of design artifact and design process data, so as to produce predictions and guidance for improving the current design process. The METRICS system has three main components: (i) the instrumentation of design tools, which includes wrapper scripts to extract data from outputs and logfiles, and callable API codes that allow direct interaction from within the design tools; (ii) the METRICS server that provides central data collection; and (iii) the data mining process that analyzes existing data to produce improvements to the existing design flow. Figure 11 shows the overall METRICS system architecture. Design and design flow data are collected by either a wrapper script or an API call from within the tools. The collected data are transmitted and stored in the METRICS server, which may reside on different machines and/or networks than those used by the design tools. Once sufficient data are collected, analyses can be performed and predictions can be made. The result of the predictions can be fed back to the design process for improvement or reporting. During collection, the data are encoded into XML format and transferred from transmitters to the web server. All data handling inside METRICS server is handled using Enterprise Java Beans. Results generated by the data miner and/or estimators are exported back to the user through servlets and applets.

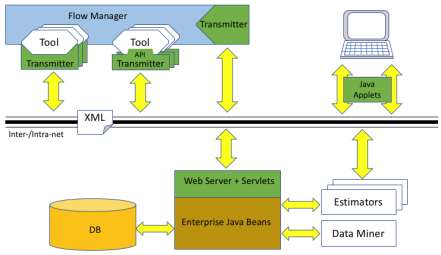

Figure 11: Overall METRICS system architecture.

Validation. METRICS was used to predict design-specific tool outcomes and best tool option settings, and to provide guidance regarding tool "sweet spots" or "field of use". Within Cadence Silicon Ensemble, tool-specific wrappers were implemented to extract data from the individual tools used in a typical SE-based flow. Multiple runs were launched with different designs and different option settings, from floorplanning to routing. Then, mining and sensitivity analyses with respect to final design QOR enabled prediction of best design-specific tool option settings. METRICS was also used to prescribe achievable clock frequency for given designs and resource budgets. The system was used to collect data from existing IP block implementations, including sizes, clock structures, timing delays, etc. During clock planning, characteristics of the IP blocks used in a given design were extracted from the database and used by an estimator to generate a target frequency range and clock structure best suited to the overall design.

Looking Back. Among the many learnings from the METRICS experience, several stand out. (1) Close collaboration and support from EDA vendors is important. Constantly changing tool behaviors and outputs are most efficiently handled through direct integration with the METRICS API - and this requires cooperation from tool developers. (2) A common METRICS vocabulary across different vendors is also important. Design metrics (crosstalk delay, vertical overcongestion, etc.) reported from one tool should have the same semantics when reported by another tool. (3) There is no "one-size-fits-all" solution for all designs. For example, data collected from designs in one technology node may not be relevant to designs in a different technology node; a rerun of data collection or other calibration may be needed. (4) Today's context is quite different from that of the original METRICS initiative.[7] (i) Reimplementing METRICS with today's commodity networking, database and cloud technologies will be much simpler compared to the initial implementation. (ii) The IC design and EDA industries are more receptive to "measure,

[6] Beyond these examples, as noted in Subsection 3.3, datasets to support ML will need to include classes of (non-infringing) artificial circuits and "eyecharts" to complement (obfuscated) real artifacts. And, the long to-do list for academic researchers includes at least (i) invention of inherently more modelable algorithms and tools (e.g., with less-chaotic behaviors than present methods) as well as (ii) collaboration toward a critical mass of open-source research infrastructure (perhaps, with similar motivations as the MARCO GSRC Bookshelf [6], and adopting the open culture seen in the AI/ML community).

[7] But, some aspects have not changed. At a DAC-2002 Birds-of-a-Feather meeting, we discovered that METRICS-like systems had been partially implemented inside several major design houses. These systems were realized using scripting languages to extract data from tools, and were largely used for resource allocation and management (i.e., licenses, machines, disk, etc.). The collected data were primarily used for reporting, with no AI deployed to process the data. Today, many companies collect LSF, flexlm and other data in Splunk [46], but usage of this data is still for "IT" rather than "design".

to improve": past resistance to "big brother watching" has been replaced by "please help me do my job". (iii) Active user interaction with the system was needed for benefits to be obtained from the predictions or flow suggestions in the METRICS data miner reports. A reimplementation of METRICS should feed predictions and guidance back into the design flow, which would then adapt tool/flow parameters midstream without human intervention.

5 CONCLUSIONS

A roadmap is traditionally defined to consist of (i) requirements (challenges) expressed as metrics, (ii) potential solutions, and (iii) a mapping between challenges and solutions – all laid out over a time horizon. By contrast, the material above gives only a personal vision – without all of the required elements of a roadmap – of how machine learning and a "METRICS 2.0" mindset may reduce time and effort in IC design. A roadmap that extends all the way to "no human in the loop" and "24-hour turnaround time" must answer such questions as the following.

(1) Are there limits to where we can reasonably seek reduction of human design effort and design cost? For example, will humans always be needed to design at the 'bleeding edge' (e.g., the latest CPU in the latest technology node)? If so, roadmapping of design time and effort reduction might chiefly address the ever-growing "fast-follower" category (e.g., 15% degradation from best possible PPA, at node (N-1)).

(2) What are appropriate metrics to track progress of time and effort reduction? For example, benchmark design tasks might be devised that map and scale to arbitrary complexities and technologies. Distinct "design driver classes" (RF, GPU, CPU, DSP, NOC, PHY) might be needed against which to measure progress.

(3) Do design productivity and design cost reflect an organization's mastery of design enablement in a given technology? This suggests the need to normalize for newness (unfamiliarity) of technology as well as a design's PPA targets relative to that technology. Additionally, metrics for IC design learning ("design capability ramp") and IC design process stability might be defined that are analogous to long-standing yield learning and process stability metrics ($D0$, Cp, Cpk) in IC manufacturing.

(4) Are "machine learning" and "sharing" compatible concepts in the design-EDA-research ecosystem? Or, will "data is the new oil" thinking induce the closing of platforms and new conflicts over the inputs and outputs of design tools and flows? And, can timely open standards preempt such behaviors?

(5) Are there fundamental inconsistencies between IC design and "big data" mindsets? [20] notes the "small data" nature of IC design, with its latency and unpredictability (IC design cannot be played out millions of times in a day as we would the game of chess), and sparsity of data (10nm layouts are much harder to find than cat images). This suggests that scalable generation of useful training data, along with standards for sharing of data and models, will be an essential element of ML-based design cost reduction.

(6) Are EDA tool predictability, stability, self-modeling, introspection, chattiness (e.g., to reveal the internal state of optimization and 'thinking') germane to a roadmap of design time and effort reduction? If so, metrics for these attributes will be needed.

6 ACKNOWLEDGMENTS

I am thankful to the many students, coauthors and collaborators whose efforts led to the works reviewed here. Sriram Venkatesh, Tushar Shah and Dr. Stefanus Mantik provided invaluable help in preparing contents of this paper. Permission of coauthors to reproduce figures is gratefully acknowledged. Research at UCSD is supported by NSF, Qualcomm, Samsung, NXP, Mentor Graphics and the C-DEN center.

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] D. Aldous and U. Vazirani, "Go With the Winners", *Proc. IEEE Symp. on Foundations of Computer Science*, 1994, pp. 492-501.
[3] S. Bansal and R. Goering, "Making 20nm Design Challenges Manageable", http://chipdesignmag.com/display.php?articleId=5144
[4] D. Bertsekas, *Dynamic Programming and Optimal Control*, Athena, 1995.
[5] K. D. Boese, A. B. Kahng and S. Muddu, "New Adaptive Multistart Techniques for Combinatorial Global Optimizations", *Operations Research Letters* 16(2) (1994), pp. 101-113.
[6] A. E. Caldwell, A. B. Kahng and I. L. Markov, "Toward CAD-IP Reuse: The MARCO GSRC Bookshelf of Fundamental CAD Algorithms", *IEEE Design and Test of Computers* 19(3) (2002), pp. 70-79.
[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, 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.
[9] 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.
[10] R. Goering, "What's Needed to "Fix" Timing Signoff?", *DAC Panel*, 2013.
[11] P. Gupta, A. B. Kahng, A. Kasibhatla and P. Sharma, "Eyecharts: Constructive Benchmarking of Gate Sizing Heuristics", *Proc. DAC*, 2010, pp. 597-602.
[12] L. Hagen and A. B. Kahng, "Combining Problem Reduction and Adaptive Multi-Start: A New Technique for Superior Iterative Partitioning", *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems* 16(7) (1997), pp. 709-717.
[13] 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.
[14] 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.
[15] K. Jeong and A. B. Kahng, "Methodology From Chaos in IC Implementation", *Proc. ISQED*, 2010, pp. 885-892.
[16] A. B. Kahng, "The Cost of Design", *IEEE Design and Test of Computers* 19(4) (2002), pp. 136-137.
[17] A. B. Kahng, "The ITRS Design Technology and System Drivers Roadmap: Process and Status", *Proc. DAC*, 2013, pp. 34-39.
[18] A. B. Kahng, DARPA IDEA Workshop presentation, Arlington, April 2017.
[19] A. B. Kahng, *keynote*, ANSYS Executive Breakfast, June 2017. http://vlsicad.ucsd.edu/Presentations/talk/Kahng-ANSYS-DACBreakfast_talk_DISTRIBUTED2.pdf
[20] A. B. Kahng, "New Directions for Learning-Based IC Design Tools and Methodologies", *Proc. ASP-DAC*, 2018, pp. 405-410.
[21] A. B. Kahng, "Quality, Schedule, and Cost: Design Technology and the Last Semiconductor Scaling Levers". *keynote*, *ASP-DAC*, 2018. http://vlsicad.ucsd.edu/ASPDAC18/ASP-DAC-2018-Keynote-Kahng-POSTED.pptx
[22] A. B. Kahng, "Machine Learning Applications in Physical Design: Recent Results and Directions", *Proc. ISPD*, 2018, pp. 68-73.
[23] A. B. Kahng and S. Kang, "Construction of Realistic Gate Sizing Benchmarks With Known Optimal Solutions", *Proc. ISPD*, 2012, pp. 153-160.
[24] 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.
[25] A. B. Kahng, S. Kumar and T. Shah, "A No-Human-in-the-Loop Methodology Toward Optimal Utilization of EDA Tools and Flows", *Proc. DAC, WIP Track*, 2018, to appear.
[26] A. B. Kahng, B. Lin and S. Nath, "High-Dimensional Metamodeling for Prediction of Clock Tree Synthesis Outcomes", *Proc. SLIP*, 2013, pp. 1-7.
[27] 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.
[28] A. B. Kahng and S. Mantik, "A System for Automatic Recording and Prediction of Design Quality Metrics", *Proc. ISQED*, 2001, pp. 81-86.
[29] A. Kahng and S. Mantik, "Measurement of Inherent Noise in EDA Tools", *Proc. ISQED*, 2002, pp. 206-211.
[30] A. B. Kahng, P. S. Purushothama, L. Saul, U. Simha and S. Venkatesh, "Timeseries Modeling for Prediction of Doomed Tool Runs", *unpublished manuscript*, 2018.
[31] A. B. Kahng and G. Smith, "A New Design Cost Model for the 2001 ITRS", *Proc. ISQED*, 2002, pp. 190-193.
[32] A. Katsioulas, S. Chow, J. Avidan and D. Fotakis, "Integrated Circuit Architecture with Standard Blocks"', *U.S. Patent* 6,467,074, 2002.
[33] B. C. May, N. Korda, A. Lee and D. S. Leslie, "Optimistic Bayesian Sampling in Contextual-bandit Problems", *J. Machine Learning Res.* (2012), pp. 2069-2106.
[34] 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.
[35] A. Olofsson, "Silicon Compilers - Version 2.0", *keynote*, *Proc. ISPD*, 2018. http://www.ispd.cc/slides/2018/k2.pdf
[36] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications on Speech Recognition", *Proc. IEEE* 77 (1989), pp. 257-286.
[37] H. Robbins, "Some Aspects of the Sequential Design of Experiments", *Bulletin of the AMS* 58(5) (1952), pp. 527-535.
[38] S. Scott, "A Modern Bayesian Look at the Multi-armed Bandit", *Applied Stochastic Models in Business and Industry* 26 (2010), pp. 639-658.
[39] G. Smith, "Updates of the ITRS Design Cost and Power Models", *Proc. ICCD*, 2014, pp. 161-165.
[40] W. R. Thompson, "On the Likelihood That One Unknown Probability Exceeds Another in View of the Evidence of Two Samples". *Biometrika* 25 (1933), pp. 285-294.
[41] *International Technology Roadmap for Semiconductors*. http://www.itrs2.net/itrs-reports.html
[42] "DARPA Rolls Out Electronics Resurgence Initiative", https://www.darpa.mil/news-events/2017-09-13
[43] The GSRC METRICS Initiative. http://vlsicad.ucsd.edu/GSRC/metrics/
[44] Partitioning- and Placement-based Intrinsic Rent Parameter Evaluation. http://vlsicad.ucsd.edu/WLD/RentCon.pdf
[45] Gate Sizing Benchmarks With Known Optimal Solution. http://vlsicad.ucsd.edu/SIZING/bench/artificial.html
[46] Splunk. https://www.splunk.com
[47] UCSD Design Cost Optimization Solver for Multi-Tapeout Project Scheduling. http://vlsicad.ucsd.edu/MILP/
[48] https://vlsicad.ucsd.edu/MAB/
[49] http://vlsicad.ucsd.edu/~abk/TALKS/japan-da-abk-final.ppt