METRICS 2.0: A Machine-Learning Based Optimization System for IC Design

(Extended abstract made available at WOSET-2018)

S. Hashemi[1], C.-T. Ho[2], A. B. Kahng[2], H.-Y. Liu[2], and S. Reda[1]
[1]*Brown University and* [2]*University of California, San Diego*
For inquires, please contact soheil_hashemi@brown.edu

***Abstract*—Despite advancements in quality of results of design automation tools, many challenges remain to be solved. In particular, there is lack of a unified and standardized system for collecting, storing, and communicating design process metrics. The lack of such system hampers the use of machine learning techniques, especially across the entire design flow. In this work, we present our current thinking toward a "METRICS 2.0" system for systematic data collection and machine learning in design automation flows. Inspired by the METRICS infrastructure, published nearly twenty years ago, we re-imagine a unified system better suited for recent developments in databases and machine learning. Our proposed system spans a wide range of components in integrated circuit design and offers (i) an updated dictionary of generic, tool and flow metrics, (ii) a standard modernized data storage approach that allows for easier data sharing, and (iii) enablement for design process outcome prediction and debugging through machine learning techniques.**

I. Introduction

In recent years, electronic design automation (EDA) tools have gained improvements in their quality of results. However, orchestrating effective data communication among tools across the design flow still requires significant time investments, especially when the design flow is composed of tools from different vendors. The lack of a unified and comprehensive design data collection and storage further reduces the feasibility of large-scale applications of machine learning (ML) in the design flow, limiting ML usage to ad-hoc attempts or narrow tool scopes [14], [15].

Nearly twenty years ago, METRICS 1.0 [2] attempted to fill this gap by providing a unified approach to collection, storage, and communication of design process parameters. With the recent advances in design automation flows, especially the open source frameworks, as well as the popularity of machine learning applications, we believe such standardization is more important than ever [8]. In this brief, we propose METRICS 2.0 a framework aimed at providing such unifying approach, with modern components better suited for current databases, design flows and machine learning tools. More specifically, our proposed METRICS 2.0 system offers a number of advanced capabilities.

- A new and updated METRICS 2.0 dictionary, where we a provide a standardize list of metrics suited for collection to characterize the design parameters and outcomes from various tools in the design flow. We amend the list published in METRICS 1.0, based on the new developments in the EDA industry in the last twenty years.
- We propose a system architecture based on JavaScript Object Notation, JSON for data logging, and mongoDB database for data storage [11] for effective storage and retrieval of the metrics. The proposed architecture eliminates the need to create database schemas and enables seamless data collection.
- We highlight the necessity of such a unified approach for enabling machine-learning approaches. Our framework is tightly coupled with machine-learning frameworks such as TensorFlow, which provides easy interfaces to read and write into MongoDB [12]. Furthermore, we describe how our framework can be utilized to enable machine learning approaches with concrete examples.

With a METRICS 2.0 infrastructure, machine learning can potentially be deployed for prediction or improvement of quality of results. For example, a model might learn to produce the best settings necessary for a tool with a given design input and design target, or to predict "doomed" runs based on initial results seen. In Section II we describe the main components of the METRICS 2.0 system. We provide the conclusions of this work in Section III.

II. METRICS 2.0

In this section we describe the components of our proposed system architecture for unified machine-learning based optimization. In Section II-D we provide a few examples where METRICS 2.0 can enable machine-learning based approaches.

A. METRICS 2.0 Dictionary

The goal of the METRICS 2.0 dictionary is to provide a comprehensive list of metrics of interest in characterizing the inputs and outcomes of a hardware design flow. Ideally, it encompasses relevant "end-to-end" metrics, starting from the designer and the design tools all the way to the final design outcomes.

The original METRICS dictionary was proposed in the late 1990s with the goal of recording all tool activities and design attributes, to enable data-mining and prediction of tool outcomes and "sweet spots". Since its inception, output tool log files and reports have changed considerably as new terminologies ("timing endpoints") and metrics (e.g., numbers

of failing endpoints, hold buffers inserted, analysis corners) have been introduced. More broadly, there is an increased interest in the use of machine learning to improve tool efficacy and reduce designer effort. Thus, collection of "big data" throughout the design process is of a more urgent need today.

In this extended abstract, we point to an initial realization of a METRICS 2.0 naming convention and metrics dictionary, which comprehends metrics for modern synthesis, placement, clock tree synthesis, routing, timing analysis and power analysis tools.[1] This initial realization includes metrics that are shared among multiple tools (e.g., "number of instances", "total instance area"); tool-specific metrics (e.g., "number of congested global cells" in the routing tool and "number of maximum capacitance violation per corner" in the timing analysis tool); as well as flow-related metrics (e.g., "tool name", "tool version" and "total runtime"). In its current form, we believe that this dictionary provides at least a minimum necessary enablement for machine learning and prediction of design flow outcomes, tool sweet spots, and design-specific tuning of tool executions. The METRICS 2.0 dictionary will be made available on GitHub [10]. We believe the WOSET community and the broader EDA community can benefit from the collaborative nature of GitHub to help improve this METRICS 2.0 dictionary into a living, "complete" set of metrics – even as collection mechanisms and machine learning methods are developed in parallel.

B. METRICS 2.0 Collection and Storage

The original METRICS 1.0 system implemented an architecture based on XML and relational database (DB) servers for storage. In METRICS 2.0 we propose the universal utilization of JSON format [5] for data collection and reporting, together with the use of NoSQL DB systems (e.g., MongoDB [11]) for data storage. More specifically, for collection of the metrics, we advocate the use of JSON-based log format in open-source design automation tools and flows. JSON [5] is a human-readable open-standard file format consisting of attribute-value pairs. As such we choose JSON as it directly maps to the structured and object-oriented nature of modern software development languages. For industrial tools, such metrics can be collected by using a wrapper around the tools to parse the required data and output JSON format. However, as the commercial tools do not provide a common APIs to collect metrics, how to effectively cope with their ever-changing logging files remains an open question.

For data storage, METRICS 2.0 relies on NoSQL MongoDB. MongoDB is a free and open-source document-based database, which offers horizontal scaling and therefore is highly scalable. In addition, MongoDB provides seamless frontend API for JSON format enabling easy interactions between the database and the design tools.

[1]Efforts toward this initial METRICS 2.0 dictionary were provided by students in UCSD CSE 249B, Fall 2018 quarter. We also acknowledge the available "report_*" Tcl commands available in modern synthesis and P&R tools, which have helped us understand how the scope of collectable tool metrics has evolved since the original METRICS dictionary [3] was developed.

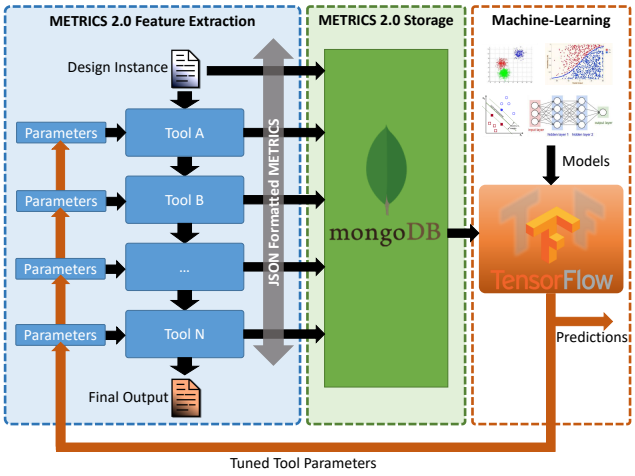

Fig. 1. The overall proposed METRICS 2.0 system architecture.

Finally, in collaboration with many open-source EDA tool developers, we are committed to a full integration of METRICS 2.0 in the open-source project OpenROAD [13], spanning over all components of hardware design.

C. Enablement for Machine-Learning

Many machine learning methodologies rely on the existence of ample training data to guide the decision-making process. METRICS 2.0 enable the large-scale collection and storage of design outcome data, by standardizing the metrics and notations across the whole hardware design stack. As a powerful database, MongoDB also provides python APIs which enables data retrieval within TensorFlow [12]. TensorFlow is an open-source library by Google targeting machine-learning and artificial intelligence applications. Written mainly in Python, TensorFlow is the most popular machine-learning framework, and enables deployment on a large range of hardware platforms, such as CPUs, GPUs, and TPUs.

Figure 1 shows the components of the proposed METRICS 2.0 architecture. Here, each tool outputs the elements of the dictionary related to that component in JSON format. The metrics are then stored in a MongoDB database, and retrieved, as necessary to enable machine-learning based optimizations. In our work, as previously described, we utilized TensorFlow, as a robust and powerful machine-learning framework, and since mongoDB provides convenient APIs for Python. Machine learning models can then be devised such that their predictions can be fed back to the design flow to guide the selection of parameters by the tools.

D. Outcome Prediction of METRICS 2.0

We now briefly discuss topics where METRICS 2.0 may be leveraged to enable machine learning approaches for process prediction and optimization. We organize this discussion according to several categories according to previous work [6], [7], and offer a few concrete examples in each category.

Logic Synthesis METRICS 2.0 should collect the delay and area metrics that result from executing logic synthesis tools under various parameter settings; i.e., logic synthesis optimization commands such as balancing, up sizing and buffering. These collections will then enable a ML approach where one can derive the best sequence of synthesis commands in order to minimize target design metric such timing. Furthermore, the collection of data from downstream tools (e.g., placement) can enable better physical synthesis optimizations that are driven by more accurate net delays instead of the general wire-load models.

Timing Analysis and Delay Calculation. METRICS 2.0 should contain extensive timing-related records, such as maximum, minimum, and average setup and hold times; setup- and hold- violating paths in different corners; maximum and minimum setup and hold time of cell instances in different corners; etc. These records can be used, e.g., to improve the efficiency of timing analysis as well as design closure. Some examples:

- Continued improvement of timing correlation and estimation along lines of [4], [9]. Matching golden analyses earlier in the flow will more accurately drive optimization and reduce ECO iterations.
- Prediction of the impact of an ECO (on setup, hold, slack, max transition, and power), across MCMM scenarios.
- Prediction of nets/paths that are more likely to be impacted by SI effects, at placement or even physical synthesis stages.

Placement and Routing. In the place-and-route arena, METRICS 2.0 should collect objects such as DRC violation per type per layer after detailed routing, total DRC violation counts after detailed routing, spatial distribution of currents, etc. Such metrics have potential to be leveraged in a wide variety of modeling and prediction tasks. For example:

- Design rule violations in particular layers, or early global routing metrics, can inform predictors of future router "failure".
- With sufficient "big data", small windows of placement and/or routing could be attacked using pattern-based approaches - e.g., using a CNN approach for detailed routing.

Routability and Power Delivery Network (PDN) Optimization. Metrics related to routing and IR drop, such as congestion hotspots, wirelength, worst-IR layers, and worst-IR instances, are useful for driving routability and PDN optimization. Examples include:

- Prediction of the convergent point for non-uniform PDN and P&R. (This would address a painful chicken-egg loop in today's methodology: PDN is defined before placement, but power analysis and routability impact can be assessed only after routing.)
- Metrics of routability and power delivery quality can be combined in useful ways. For example, [1] describes a power delivery *pathfinding* methodology for emerging die-to-wafer integration, wherein ML models are used to predict near-optimal PDN solutions for given design and PDN specification.

Power and IR Drop Analysis. METRICS 2.0 should capture additional power- and IR-related characteristics, such as worst static/dynamic IR drop across all instances, switching power, internal power, (vectorless) dynamic IR drop, and instance toggle rates. Such power and IR metrics, along with instance information collected at the synthesis stage (number of buffers, clock gating instances, and sequential and combinational instances, etc.), can inform early predictions of power and IR drop analyses. For example:

- Switching activity prediction for power and IR drop analysis. With the IR and Power records of different functional design structure and scenarios, we may be able to predict or model the switching activity across scenarios and ensure sufficient coverage of scenarios.
- Prediction of IR drop hotspots, worst IR value, and potential EM hotspots across design stages. Leveraging the worst IR drop of instances, pad current, EM hotspots, and instance power records to predict the IR/EM hotspots of different PDN designs.

Resource Management. Last, to optimize the design process itself, infrastructure and configuration must be taken into account. Even with identical parameter setup, if the compute resource varies across flow executions, the final results will differ. Thus, like its predecessor, METRICS 2.0 should include flow-related metrics: tools and versions used, memory usage, server and runtime information. These metrics can enable predictive modeling aimed at maximizing productive tool utilization or minimizing overall design (engineer, license, compute) resource cost.

- Prediction of runtime and memory requirements is still a basic open issue at all stages of the design process.
- Metrics related to area, power, and slack distribution of the design (e.g., collected from synthesis) can be used to predict the runtime and (distributions of) outcome of physical implementation.

This wide range of examples suggests that realizing the full potential of machine-learning based approaches in hardware design automation is only possible with large-scale, pervasive collection of relevant data. To this end, our envisioned METRICS 2.0 can offer a unified, turnkey platform for future collection and storage of design process information throughout the entire design flow.

III. Conclusions

In this work, we have proposed METRICS 2.0 to enable systematic data collection and analysis that better fit the needs and capabilities of today's EDA tools. METRICS 2.0 relies on JSON as the intermediate data representation and MongoDB for scalable data storage. Our unified approach enables large-scale and standardized collection and storage of data outcome from hardware design flows. Such unified metrics can then be utilized for developing machine-learning based methodologies

that can predict the output of the design flow, or provide feedback on the tool parameters to improve the overall quality of results. We aim at integrating METRICS 2.0 within an end-to-end hardware design process as part of the OpenROAD project [13] .

REFERENCES

[1] S. Kim A. B. Kahng, S. Kang and B. Xu. Power Delivery Pathfinding for Emerging Die-to-Wafer Integration Technology. In *Proc. DATE*, 2019, to appear.

[2] S. Fenstermaker, D. George, A. B. Kahng, S. Mantik, and B. Thielges. METRICS: A System Architecture for Design Process Optimization. In *Proc. DAC*, pages 705–710, 2000.

[3] S. Fenstermaker, D. George, A. B. Kahng, S. Mantik, and B. Thielges. METRICS: A System Architecture for Design Process Optimization. In *Proc. DAC*, pages 705–710, 2000.

[4] S.-S. Han, A. B. Kahng, S. Nath, and A. S. Vydyanathan. A Deep Learning Methodology to Proliferate Golden Signoff Timing. In *Proc. DATE*, 2014.

[5] JSON: JavaScript Object Notation. https://www.json.org/.

[6] A. B Kahng. Machine Learning Applications in Physical Design: Recent Results and Directions. In *Proc. ISPD*, pages 68–73, 2018.

[7] A. B Kahng. New Directions for Learning-Based IC Design Tools and Methodologies. In *Proc. ASP-DAC*, pages 405–410, 2018.

[8] A. B. Kahng. Reducing Time and Effort in IC Implementation: A Roadmap of Challenges and Solutions. In *Proc. DAC*, pages 1–36, 2018.

[9] A. B. Kahng, M. Luo, and S. Nath. SI for Free: Machine Learning of Interconnect Coupling Delay and Transition Effects. In *Proc. SLIP*, pages 1–8, 2015.

[10] METRICS 2.0. https://github.com/The-OpenROAD-Project/METRICS-2.0.

[11] MongoDB. https://www.mongodb.com/.

[12] TensorFlow: An Open Source Machine Learning Framework for Everyone. https://tensorflow.org.

[13] The OpenROAD Project. https://theopenroadproject.org/.

[14] C. Yu, H. Xiao, and G. De Micheli. Developing Synthesis Flows Without Human Knowledge. In *Proc. DAC*, pages 1–6, 2018.

[15] M. M. Ziegler, H. Liu, G. Gristede, B. Owens, R. Nigaglioni, and L. P. Carloni. A Synthesis-Parameter Tuning System for Autonomous Design-Space Exploration. In *Proc. DATE*, pages 1148–1151, 2016.