DATC RDF-2020: Strengthening the Foundation for Academic Research in IC Physical Design

Invited Paper

Jianli Chen[1], Iris Hui-Ru Jiang[2], Jinwook Jung[3], Andrew B. Kahng[4], Victor N. Kravets[3], Yih-Lang Li[5], Shih-Ting Lin[5], and Mingyu Woo[4]

[1]Fudan University, Shanghai, China
[2]National Taiwan University, Taipei, Taiwan
[3]IBM T. J. Watson Research Center, Yorktown Heights, NY, USA
[4]UC San Diego, La Jolla, CA, USA
[5]National Chiao Tung University, Hsinchu, Taiwan

ABSTRACT

We describe the *RDF-2020* release of the IEEE CEDA DATC Robust Design Flow (RDF). RDF-2020 extends the previous four years of DATC efforts to (i) preserve and integrate leading research codes, including from past academic contests, and (ii) provide a foundation and backplane for academic research in the RTL-to-GDS IC implementation arena. Implementation and analysis flows have been enhanced by the addition of steps including multi-bit flip-flop clustering, parasitic extraction and antenna checking, as well as a recent contest-winning global router. RDF-2020 also opens a new "Calibrations" direction to support academic research on key analyses such as extraction and timing. An open-source physical design database with Tcl/Python/C++ APIs, a flow integration into a single scriptable application, and support for the newly-opened SKY130 manufacturable PDK, are also new this year. Our paper closes with a discussion of potential future directions for the RDF effort.

KEYWORDS

RTL-to-GDS, academic reference flow, open source, VLSI CAD, electronic design automation.

1 INTRODUCTION

IEEE CEDA's Design Automation Technical Committee (DATC) [1] has over the past five years developed the *DATC Robust Design Flow* (RDF), which aims to facilitate academic research on flow-scale methodology and cross-stage optimizations in the physical IC implementation ("RTL-to-GDS") domain. The RDF mission is to preserve and integrate leading research codes, including a number of winning entries from past academic contests, while also establishing a robust foundation for academic RTL-to-GDS research. A series of papers beginning at ICCAD-2016 [2–6] documents the progress of the DATC RDF toward (i) providing an academic reference flow from logic synthesis to detailed routing based on existing contest results; (ii) curating a collection of design benchmarks and point tool libraries; and (iii) connecting academic research to industrial practice and designs by fully supporting industry-standard design interchange formats.

A year ago, the focus in RDF-2019 was on *vertical and horizontal extensions* to achieve a complete flow with multiple options available within the tool chain. Key advances included (i) completion of a full RTL-to-GDS flow, (ii) the first RDF-based tool chain comprised entirely of open-source components, and (iii) support for a full set of standard (as opposed to "translated" or "interpreted") industry design interchange, library and constraint formats [6].

In the past year, *RDF-2020* efforts have filled in and solidified the RTL-to-GDS implementation flow, adding intermediate flow steps as well as new tool options. The scope and mission of RDF has also been updated, bringing new attention to the support of analysis and verification research, and more clearly documenting a roadmap of RDF needs and planned enhancements. As detailed below, RDF's implementation and analysis flows have been enhanced by the addition of steps including multi-bit flip-flop clustering, parasitic extraction and antenna checking, as well as a recent contest-winning global router. RDF-2020 includes a new "Calibrations" direction to support academic research on key analyses such as parasitic extraction and static timing. An open-source physical design database with Tcl/Python/C++ APIs, a flow integration into a single scriptable EDA application, and support for the newly-opened SKY130 manufacturable PDK [16], are also new this year.

In the following, Section 2 reviews the current status of RDF-2020, focusing on several key updates. Section 3 describes the extraction and STA calibrations that have been added this year. Last, Section 4 discusses potential futures, including further extensions of RDF and evolutions of the RDF mission.

2 RDF-2020: UPDATES AND STATUS

In this section, we summarize the key "deltas" seen in RDF-2020. These include the following.

- A recently open-sourced *foundry* PDK, **SkyWater 130nm** (SKY130), is now supported in RDF.
- The integrated database in the OpenROAD project, **OpenDB**, with in-built Tcl, Python and C++ APIs, is added to RDF. This

ICCAD '20, November 2–5, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-8026-3/20/11…$15.00
https://doi.org/10.1145/3400302.3415742

Table 1: Overview of the RDF-2020 flow

Component	Tool
Logic synthesis	Yosys+ABC
Floorplanning	TritonFP
Global placement	RePlAce, FZUplace, NTUPlace3, ComPLx, Eh?Placer, FastPlace3-GP, mPL5/6, Capo
Detailed placement	OpenDP, MCHL, FastPlace3-DP
Flip-flop clustering	**Mean-shift**, **FlopTray**
Clock tree synthesis	TritonCTS
Global routing	FastRoute4-lefdef, NCTUgr, **CUGR**
Detailed routing	TritonRoute, NCTUdr, DrCU
Layout finishing	**KLayout**, Magic
Gate sizing	Resizer, TritonSizer
Parasitic extraction	**OpenRCX**
STA	OpenSTA, iTimerC
Database	**OpenDB**
Libraries/PDK	NanGate45, **SKY130**, ASAP7, NCTUcell
(integrated app)	**(OpenROAD app)**

eases the task of supporting and integrating into flow studies numerous past works in the field.

- There has been continued growth of the set of available point tools or engines, with new additions including the ICCAD-2019 Contest winning global router (CUGR), an antenna-checker, and two multi-bit flip-flop clustering codes.
- KLayout has been added as a more scalable alternative to Magic in the layout finishing stage of the RDF flow.

Table 1 summarizes the RDF-2020 flow, highlighting the newly-included components in boldface.

2.1 Foundry PDK and Enablement

RDF-2019 was previously tested on NanGate45 [22] and ASAP7 [21] PDK and libraries. Neither platform corresponds to a manufacturable technology. Furthermore, the ASAP7 license currently prevents download by commercial entities,[1] so only the [22] enablement has been in general use. In this light, it is highly significant that a foundry-manufacturable 130nm PDK and design enablement (including cell libraries, IOs, RAM generator, along with signoff and physical verification kit) was open-sourced by Google and SkyWater Technology Foundry earlier this year [16]. Below, we describe RDF-2020 support and example results on the three enablements [16, 21, 22].

2.2 NCTUcell Standard Cell Library

Multi-bit flip-flop (MBFF) is a key physical design lever used to reduce power without large performance degradation. In previous cell libraries used with RDF, no MBFF cells are available. In RDF-2020, NCTUcell [11] has been successfully used to synthesize 2-bit and 4-bit MBFF cells – which are the most commonly-used MBFF sizes – on the ASAP7 and NanGate45 (FreePDK45) PDKs. ASAP7 is a FinFET technology while FreePDK45 is a planar transistor technology; there are many differences between the design rules of these PDKs. One major distinction is that uniform and aligned gates are required in ASAP7, but are not necessary in FreePDK45.

[1]We hope that this will change before publication of our paper.

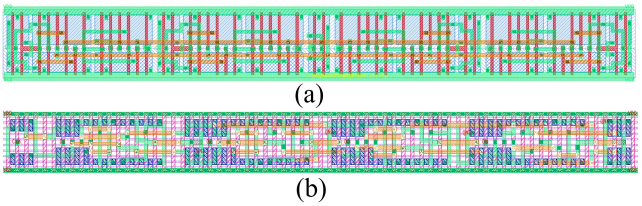


Figure 1: Synthesized 4-bit MBFFs for (a) FreePDK45 and (b) ASAP7.

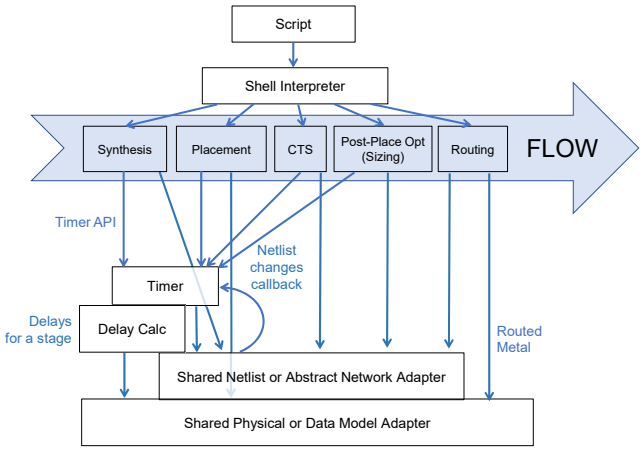


Figure 2: Incremental shared netlist structure in OpenDB [17]. The thin blue arrows illustrate the function or API calls and the thick blue arrow illustrates the overall flow from logic synthesis through detailed routing.

A second distinction is that poly is an available routing layer in FreePDK45, but this is not the case in ASAP7. The NCTUcell tool was initially designed for FinFET technologies. With respect to the first distinction above, NCTUcell can satisfy requirements of the design rule set in FreePDK45, but is not capable of optimizing non-uniform gate placement. With respect to the second distinction, FreePDK45 allows use of the poly layer to connect continuous gates, and ASAP7 has a related M0 layer, called LIG, for the same purpose. Thus, the existing routing utility in NCTUcell can be applied to handle the design rule set of FreePDK45. Figure 1 shows the layouts of 4-bit flip-flop (FF) cells for FreePDK45 and ASAP7.

2.3 OpenDB Database

OpenDB [18] [17] is a new open-source physical implementation database that holds all essential data for the physical design creation flow (floorplan, global and detailed placement, CTS, and global and detailed routing) as well as timing and power analyses. The underlying data model of OpenDB is similar to that of the LEF/DEF exchange formats, or the well-known OpenAccess database [24]. A significant current limitation is that OpenDB has a flattened netlist structure; implementation of a hierarchical netlist structure is an important future extension.

Figure 2 shows the highly incremental, shared netlist architecture of a modern back-end EDA tool. OpenDB underlies this picture.

The open-sourced timer, OpenSTA [27], is intimately connected with OpenDB such that both timing graph and physical design information are accessible to tools such as a sizing optimizer. The shared netlist data structure enables in-memory communication between tools and the speed improvements that make tight incremental optimization loops feasible. Since the release of OpenDB, nearly 20 distinct projects have now been integrated into a single binary, referred to as the **OpenROAD top-level app**. Redundancies and inconsistencies such as multiple LEF/DEF readers and writers, as well as file-based or name-based communication between flow steps, have been eradicated. Instead, all projects utilize OpenDB's data structure, via C++ and Tcl APIs. Note that our latest RDF-2020 *contains* the OpenROAD top-level app, which is built on OpenDB. We have added wrappers that enable multiple flip-flop clustering tools' outputs (see below) to be properly updated into OpenDB.

In a recent DAC-2020 tutorial [30], several examples of OpenDB usage are provided as examples of using OpenDB Tcl and C++ APIs. The `pdngen` and `tapcell` applications are part of the OpenROAD top-level app and use the OpenDB Tcl API exclusively. The `ClipGraphExtract` demonstrates how to add a tool into the OpenROAD top-level app using the OpenDB C++ API. `ClipGraphExtract` takes LEF/DEF and clip coordinates, and generates the edge list (e.g., an instance-instance pair) within the given clip coordinates. The edge list can be part of a dataset used to train, e.g., graph neural networks (GNN) or graph convolution network (GCN) on clip-wise predictions.

2.4 Post-Placement Flip-Flop Clustering

RDF-2020 flow includes an MBFF clustering stage after placement. It clusters flip-flops in the given placement and generates MBFF-mapped netlist and DEF. The goal is to minimize clock power by reducing the number of clock sinks and thereby the total sink pin capacitance. The clustering can be enabled or disabled via flow configuration. The following two flip-flop clustering algorithms have been added into the RDF-2020 flow.

2.4.1 Mean-Shift Clustering. Given a timing-optimized placement, creating large clusters or dragging outliers far away inevitably causes large disruption to placement thus incurring significant timing degradation and timing ECO efforts.

For reducing clock power while minimizing timing degradation, *effective mean shift* naturally forms clusters according to flip-flop distribution without placement disruption [32, 33]. By augmenting classic mean shift algorithm with special treatments, effective mean shift fulfills the requirements to be a good flip-flop clustering algorithm because it needs no pre-specified number of clusters, is insensitive to initialization, is robust to outliers, is tolerant of various register distributions, is efficient and scalable, and balances clock power reduction against timing degradation.

2.4.2 Flop-Tray Clustering. The "flop-tray" clustering method of [31] uses combinatorial methods to solve the "chicken-and-egg" loop between flop-tray generation (i.e., MBFF clustering) and placement optimization, and to achieve improved solution quality especially when a range of MBFF sizes is available. For each given MBFF size (e.g., 4-bit, 8-bit, etc.), capacitated K-means clustering (min-cost flow) is used to cluster the FFs, and linear programming is used to determine best locations for the MBFF instances that correspond to FF clusters, taking into account the footprint (aspect ratio) of the MBFF. An integer linear program is then used to determine the combination of MBFF sizes and placements that minimizes FF displacements, timing impacts and number of isolated sinks.

2.5 Global and Detailed Routing

CUGR [36] was the winning entry in the ICCAD 2019 Open-Source LEF/DEF Based Global Routing Contest that was organized by Mentor, Cadence and UCSD [38]. CUGR has been open-sourced by its authors [37], and is another welcome instance (complementing FastRoute4-lefdef) of the bridge between academic placement and detailed routing that can be run in a full industry enablement.

CUGR performs two techniques: 3D pattern routing and multi-level 3D maze routing. The 3D pattern routing combines pattern routing and layer assignment. The multi-level 3D maze routing has different cost functions on each level, enabling efficient searches for best-cost routes. CUGR reads in LEF/DEF using Rsyn [41], which could in future be evolved to use the LEF/DEF readers in OpenDB/OpenROAD. Currently, RDF-2020 can utilize the CUGR binary to run the global routing in the flow.

2.6 Other Additions

Several other noteworthy additions to RDF-2020, via the OpenROAD project, are the following.

KLayout [23] is used for layout finishing, dummy fill, streamout and other tapeout-related functions. In particular, within the RTL-to-GDS flow, it produces the final GDS from the DEF layouts and the GDS of design instances. Dummy metal fill has also been implemented in KLayout, although this functionality may soon migrate from KLayout to a standalone dummy fill generator.

OpenRCX [19] is an open-source 2.5-D parasitic extraction developed by a mid-2000s EDA startup, Athena Design Systems, Inc. Initially, the extraction engine was developed and supported up through customer engagements at 90nm technology nodes. Since the code was open-sourced in Dec. 2019, it has been improved to support current commercial FinFET technologies. OpenRCX is integrated into the OpenROAD top-level app, and is also separately available in a public repository [19]. Calibrations versus foundry technologies have been performed on the FreePDK45 and SKY130.

ARC [20] is an open-source antenna rull checker running on global or detailed routed designs. It supports PAR (partial area ratio check) and a subset of CAR (cumulative area ratio check) types antenna ratio rules. PAR rules pertain to any single metallization step, while CAR rules take into account a current metal layer and the layers below it.

Given post-route information in OpenDB, ARC generates an undirected wire-graph where the nodes represent elements of metal interconnections and the edges represent connections in the wire (VIAs, METALs). ARC can generate a report to indicate nets that violate antenna rules. APIs are provided as an interface to preemptively fix antenna violations (e.g., by bridging in an incremental global routing step) or to accomplish diode insertion (which invokes incremental netlist update, placement legalization and global routing). ARC was introduced as a tutorial example for the OpenROAD top-level app's Tcl API in [30].

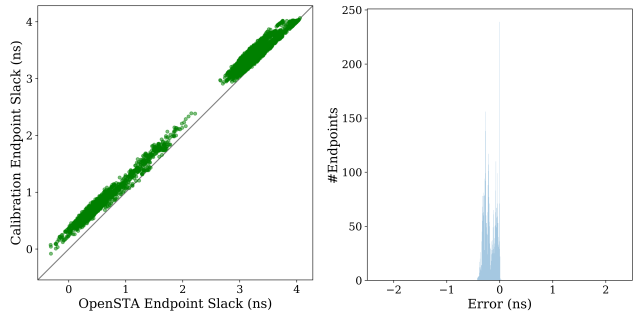


Figure 3: Correlation analysis results on *jpeg_encoder* with NanGate45. (a) Endpoint slack comparison between OpenSTA [27] and commercial timing analysis results. (b) Error distribution from (a).

3 A NEW DIRECTION: ANALYSIS AND VERIFICATION

An important direction for the evolution of the DATC RDF is to close academic research enablement gaps in the *analysis and verification* arena. For example, RDF does not have any tools to perform DRC/LVS checking, and this is not an active area of academic research. In this section, we first discuss a gap that is newly addressed in RDF-2020, namely, that academic research on parasitic extraction and static timing analysis is *uncalibrated*: the tools accept inputs and generate outputs, but there are no standard calibration datasets against which improvements can steadily be made. We also discuss additional gaps and calibration opportunities for analysis and verification, notably signal integrity, power integrity and DRC/LVS checking.

3.1 Calibrations

In the past year, RDF-2020 has added *calibrations* for two basic electrical analyses – parasitic estimation and static timing analysis – using publicly-available enablements (NanGate45, SKY130). Our hope is that these calibration datasets will help boost the research community's advancement along the axes of accuracy, turnaround time, and capacity for these fundamental analyses that inform IC physical implementation. We also believe that the DATC RDF can provide a repository for an ever-growing collection of such analysis calibration data.

With respect to calibration of timing analysis, the calibration data can begin with Verilog (.v), timing constraints (.sdc), routed .def, and an extracted .spef.[2] From these inputs, golden calibration results of static timing analysis can be compiled from any source timing analysis tool.

Our initial data compilation uses four DRV-free routed DEFs produced by the OpenROAD flow: aes_cipher_top and jpeg_encoder designs [26], in each of the SKY130 [16] and NanGate45 [22] enablements. Golden calibration data is abstracted and anonymized using

[2]The .spef is itself taken from our calibration data for parasitic extraction, which is edited (and, which is optionally obfuscated by small zero-mean perturbations) to anonymize any source extraction tool.

```
============================================================
  aes_cipher_top (freepdk45) Summary
============================================================
WNS: -0.230
TNS: -10.560
FEP: 139
------------------------------------------------------------
  aes_cipher_top (freepdk45) top1 worst timing path
------------------------------------------------------------
Startpoint: _28827_/Q (Falling)
Endpoint: _28884_/D (Rising)
Path Group: reg2reg

  Delay    Time   Description
------------------------------------------------------------
   0.00    0.00 ^ clk
   0.01    0.01 ^ clkbuf_0_clk/A (BUF_X4)
   0.03    0.04 ^ clkbuf_0_clk/Z (BUF_X4)
...
   0.00    1.54 ^ _28884_/D (DFF_X1)
           1.54   data arrival time

   0.00    1.00 ^ clk
   0.01    1.01 ^ clkbuf_0_clk/A (BUF_X4)
   0.03    1.03 ^ clkbuf_0_clk/Z (BUF_X4)
...
  -0.04    1.31   library setup time
           1.31   data required time
------------------------------------------------------------
           1.31   data required time
           1.54   data arrival time
------------------------------------------------------------
          -0.23   slack (VIOLATED)
```

Figure 4: Example timing analysis calibration data. The timing report viewer produces this OpenSTA-style [27] format from the 5-worst JSON.

a *5-worst JSON* format, which we use to hold block-level worst (negative) slack, total negative slack, and number of failing endpoints (i.e., standard WNS, TNS and FEP metrics), along with detailed information for the top-5 worst timing paths (including arc delays and pin arrival times). We provide a timing report viewer that reads 5-worst JSON-formatted data and prints out a timing report in the OpenSTA tool's [27] report format, as shown in Figure 4.

To facilitate other calibrations of interest, we also propose an *endpoints JSON* format, which can capture setup slack values at every flip-flop D pin. As shown in Figure 3, we can compare the endpoint slacks from OpenSTA [27] with calibration endpoint slack values (in the plot shown, the calibration values are obtained with signal integrity option disabled). For these examples, all Verilog, DEF, SPEF, SDC, 5-worst JSON, endpoints JSON, and timing report viewer are open-sourced in [29].

3.2 Further Calibrations

In the electrical analysis and verification area, RDF-2020 calibrations do not yet touch signal integrity, power integrity or power analyses. In future iterations of RDF, we plan to add analysis calibrations according to the sequence: (i) static IR drop map, (ii) vectorless dynamic power, and (iii) vectorless dynamic IR drop map. Our current strategy of mapping information from source analysis reports to a generic JSON format, adding obfuscations (e.g., random small perturbations) as needed, readily extends to these future calibrations.

Near-term extensions of our initial RCX and STA calibrations include (i) signal integrity analysis in STA, and (ii) current source model-based delay calculation. The latter may build on the recent TAU-2020 Contest [39]. We note that data collected for any of the above calibrations can *also* serve as a training / testing foundation for development of separate *machine learning* capabilities, e.g., to

shift the cost-vs.-accuracy tradeoff in performance estimation, or to improve design closure. Other opportunities for machine learning in the RDF context are discussed in Section 4.3 below.[3]

4 FUTURE SCOPE: RESEARCH CHALLENGES AND EXTENSIONS

RDF-2020's coverage of the RTL-to-GDS implementation flow largely satisfies the original RDF goal. Of course, functionality and solution quality should continue to be improved on both the implementation and analysis/verification sides of RDF. At the same time, there is now an opportunity to begin evolving RDF *less reactively* (e.g., adding new tools as a function of the contests held in the previous year by major conferences), and *more proactively*. In this section, we outline several research challenges and extensions that are of particular interest for RDF's future.

4.1 Non-Integer Multiple-Height Cells

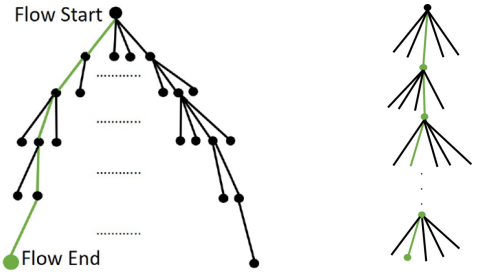
Typical standard cell libraries provide multiple standard cell architectures, e.g., smaller track heights (6T/8T) for high-density designs, and larger heights (7.5T/12T) for high-speed designs. Designers usually adopt a single type for ease of design, based on the design objective. However, leveraging the *non-integer multiple-height* (*NIMH*) cells entirely can give more room to optimize PPA, providing larger solution space and so better QoR. Dobre *et al.* [35] showed that, in a 28nm industrial node, using both 12T and 8T cells in a circuit can reduce the total cell area by 25% without sacrificing performance, compared to using only 12T cells. Compared to 8T implementation, NIMH improved performance by 20% with a similar area. Nevertheless, few works address NIMH design methodology from synthesis through placement and routing.

4.2 Logic Locking for Hardware Security

During the past decade, hardware security research has produced a broad portfolio of approaches to counteract piracy of semiconductor chips. *Logic locking* has emerged as a particularly appealing technique that focuses on preventing unauthorized use of hardware intellectual property (IP). The method hides the oracle behavior of a design by instrumenting a logic that encrypts its core functionality. The activation of a fabricated chip's correct function requires a digital key, shared by the IP holder. The evolving research in the field spans multiple vectors: it iterates over devising locking techniques [42], assessing their resilience to an attack model [43, 44], and responding with attack-mitigating remedies [45, 46].

Due to the diverse and incremental nature of the developed approaches, the progress in the field would benefit from a common platform that reconciles assumptions and simplifies sharing of open-source. For example, reverse engineering logic camouflaged with light-weight synthesis [47] may over-simplify the attacking barriers; logic post-processed with the aggressive circuit restructuring of DATC RDF would present an additional overhead to restore the original representation of the embedded locked mechanism. The relevance of assumptions may also be vulnerable to the expanding scope of state-of-the-art design automation. For example, the assumed limitation that oracle design behavior is available to an intruder only in the observational form of sample responses – and not as the explicit representation of its function – may not hold over time. The upcoming advances can render such a restriction too conservative, as the learning of a Boolean function from input-output pairs becomes more feasible [48].

[3]Here, we do not discuss physical verifications such as design rule checking (DRC), layout versus schematic checking (LVS), or related functions such as lithography simulation. However, in the past these have been the subject of academic contests and very active academic research, and the collection of calibration data to support academic researchers may be of interest.



(a) (b)

Figure 5: (a) Tree of flow stages, and options at each stage. Each leaf in the tree is a potential flow outcome. (b) A "smart flow" might apply the go-with-the-winners paradigm, where most-promising solutions at a given flow stage are cloned and passed on to the next flow stage. (Source: [12])

4.3 A "Smart" RDF Flow

Having a complete flow, with multiple tool options and "recipes" at each flow stage, begs the question: How should RDF be best applied to meet a given (performance, power, area density) target for a given design? This gives rise to a "*smart* RDF flow" challenge. Figure 5(a), reproduced from [12], cartoons the tree of potential flow-stage recipes and eventual flow outcomes. A simple smart RDF flow might follow the "go-with-the-winners" (GWTW) strategy [13], launching multiple optimization threads, and periodically identifying and cloning the most promising thread(s) while terminating other threads (Figure 5(b)). The (machine learning) research challenge is to learn how to identify "most promising" states of the design at key junctures (floorplan, global placement, post-CTS, post-global route). Reinforcement learning (e.g., multi-armed bandit methods [40]) might also be used to develop a smart RDF flow.

4.4 Further Challenges and Extensions

Beyond the directions listed above, DATC members have compiled the following list of additional opportunities. We look forward to prioritizing or even incentivizing these improvements to RDF, in collaboration with the EDA community.

- DFT support is currently missing from RDF. A routability- and/or timing-aware multiple scan chain ordering engine [14] can be one initial target, possibly building on the recently-released Fault package [15].
- Clock gating synthesis is also missing. Currently, no public logic synthesizer/optimizer can generate clock gating, although it is a standard, pervasive design practice to minimize clock switching power.

- Gate-level vector-based power estimation is an important analysis for which both tools and calibration data might be added to the RDF flow, following the trajectory of Subsection 3.2 above.
- Several fundamental optimizations are currently "covered" in RDF-2020 but with much of room left for QoR improvement. Clock tree synthesis (where clock skew scheduling is currently missing) and gate sizing (where multi-corner sizing optimization is currently missing) are two examples.
- Combining sizing / VT-swap with (incremental) routing *and* cell movement (cf. the ICCAD-2020 Contest Problem B [28]) is a key challenge for advanced-node physical design that is now more accessible with the OpenDB-based incremental shared netlist structure.
- Finally, a longer-term goal is to support more open, standardized *modeling* frameworks to support physical design research. Domains such as device modeling are well-served by industry consortia, with numerous public standards (BSIM, PSP, etc.). However, areas such as interconnect parasitic modeling, OPC (resist and source) modeling, or CMP (planarization) modeling have no available standard, which blocks both research and research-industry collaborations.

5 CONCLUSION

Several significant developments from the past year are seen in RDF-2020. These include flow integration onto a freely distributable backplane of database and timer, with an incremental shared netlist architecture; bringup of the RDF flow onto the manufacturable SKY130 PDK; initiation of support for analysis *calibrations*; and further horizontal and vertical additions to RDF. We also describe several research challenges now addressable by the research community with RDF-2020, along with potential future extensions. RDF will explore these and other directions of growth as it continues to improve its support of academic physical design research.

6 ACKNOWLEDGMENTS

RDF is an initiative of the IEEE CEDA Design Automation Technical Committee (DATC). We thank the IEEE CEDA executive committee, particularly Dr. Gi-Joon Nam of IBM, for their support.

REFERENCES

[1] "IEEE CEDA Design Automation Technical Committee," https://ieee-ceda.org/node/2591.
[2] J. Jung, I. H.-R. Jiang, G.-J. Nam, V. N. Kravets, L. Behjat, and Y.-L. Li, "OpenDesign Flow Database: The infrastructure for VLSI design and design automation research," *Proc. ICCAD*, Nov. 2016, pp. 42:1–42:6.
[3] J. Jung, P.-Y. Lee, Y. Wu, N. K. Darav, I. H. Jiang, V. N. Kravets, L. Behjat, Y. Li, and G. Nam, "DATC RDF: Robust design flow database," *Proc. ICCAD*, Nov. 2017, pp. 872–873.
[4] J. Jung, I. H.-R. Jiang, J. Chen, S.-T. Lin, Y.-L. Li, V. N. Kravets, and G.-J. Nam, "DATC RDF: An academic flow from logic synthesis to detailed routing," *Proc. ICCAD*, Nov. 2018, pp. 37:1–37:4.
[5] —, "DATC RDF: An open design flow from logic synthesis to detailed routing," *Proc. Workshop on Open-Source EDA Technology (WOSET)*, Nov. 2018, pp. 6:1–6:4.
[6] J. Chen, I. H.-R. Jiang, J. Jung, A. B. Kahng, V. N. Kravets, Y.-L. Li, S.-T. Lin and M. Woo, "DATC RDF-2019: Towards a complete academic reference design flow," *Proc. ICCAD*, Nov. 2019, pp. 1–6.
[7] T. Ajayi, V. A. Chhabria, M. Fogaca, S. Hashemi, A. Hosny, A. B. Kahng, M. Kim, J. Lee, U. Mallappa, M. Neseem, G. Pradipta, S. Reda, M. Saligane, S. S. Sapatnekar, C. Sechen, M. Shalan, W. Swartz, L. Wang, Z. Wang, M. Woo, and B. Xu, "Toward an open-source digital flow: First learnings from the OpenROAD project," *Proc. DAC*, June 2019, pp. 76:1–76:4.
[8] "The OpenROAD Project," https://github.com/The-OpenROAD-Project.
[9] "IEEE CEDA DATC Robust Design Flow," https://github.com/ieee-ceda-datc/datc-robust-design-flow.
[10] "NanGate FreePDK45 Generic Open Cell Library," http://projects.si2.org/openeda.si2.org/projects/nangatelib.
[11] Y.-L. Li, S.-T. Lin, S. Nishizawa, H.-Y. Su, M.-J. Fong, O. Chen, and H. Onodera, "NCTUcell: A DDA-aware cell library generator for FinFET structure with implicitly adjustable grid map," *Proc. DAC*, June 2019, pp. 1–6.
[12] A. B. Kahng, "Reducing time and effort in IC implementation: A roadmap of challenges and solutions," *Proc. DAC*, June 2018, pp. 36:1–36:6.
[13] D. Aldous and U. Vazirani, ""Go with the winners" algorithms," *Proc. IEEE Symp. on Foundations of Computer Science*, 1994, pp. 492–501.
[14] A. B. Kahng, I. Kang and S. Nath, "Incremental multiple-scan chain ordering for ECO flip-flop insertion," *Proc. ICCAD*, Nov. 2013, pp. 705–712.
[15] "Fault," https://github.com/Cloud-V/Fault.
[16] "SkyWater Open Source PDK," https://github.com/google/skywater-pdk.
[17] T. Spyrou, "OpenDB, OpenROAD's Database," *Proc. Workshop on Open-Source EDA Technology (WOSET)*, Nov. 2019, pp. 6:1–6:2.
[18] "OpenDB," https://github.com/The-OpenROAD-Project/OpenDB.
[19] "OpenRCX," https://github.com/The-OpenROAD-Project/OpenRCX.
[20] "Antenna Rule Checker," https://github.com/The-OpenROAD-Project/OpenROAD/tree/openroad/src/antennachecker.
[21] L. T. Clark, V. Vashishtha, L. Shifren, A. Gujja, S. Sinha, B. Cline, C. Ramamurthy, and G. Yeric, "ASAP7: A 7-nm FinFET predictive process design kit," *Microelectronics J.* 53, pp. 105–115, July 2016.
[22] "NanGate FreePDK45 Generic Open Cell Library," http://projects.si2.org/openeda.si2.org/projects/nangatelib.
[23] "KLayout," https://www.klayout.de/.
[24] "OpenAccess Silicon Integration Initiative," https://si2.org/openaccess/.
[25] "OpenLANE," https://github.com/efabless/openlane.
[26] "OpenCores," https://opencores.org/.
[27] "OpenSTA," https://github.com/The-OpenROAD-Project/OpenSTA.
[28] "Routing With Cell Movement," *ICCAD-2020 CAD Contest*, Problem B. http://iccad-contest.org/2020/problems.html.
[29] "DATC RDF Timer Calibration," https://github.com/ieee-ceda-datc/datc-rdf-timer-calibration.
[30] "Developing Industrial Strength EDA Tools Using the OpenDB Open-source Database and the OpenROAD Framework," https://github.com/The-OpenROAD-Project/DAC-2020-Tutorial.
[31] A. B. Kahng, J. Li, and L. Wang, "Improved flop tray-based design implementation for power reduction," *Proc. ICCAD*, Nov. 2018, pp. 1–8.
[32] Y. Chang, T.-W. Lin, I. H.-R. Jiang, and G. Nam, "Graceful register clustering by effective mean shift algorithm for power and timing balancing," *Proc. ISPD*, Apr, 2019, pp. 11–18.
[33] "Effective Mean Shift Latch Clustering," https://github.com/waynelin567/Register_Clustering.
[34] M. Hatamian and P. Penzes, "Non-integer height standard cell library," *U.S. Patent 8,788,998*, July 2014.
[35] S. A. Dobre, A. B. Kahng, and J. Li, "Design implementation with noninteger multiple-height cells for improved design quality in advanced nodes," *IEEE TCAD*, 37(4), pp. 855–868, Apr. 2018.
[36] J. Liu, C.-W. Pui, F. Wang and E. F. Y. Young, "CUGR: Detailed-routability-driven 3D global routing with probabilistic resource model," *Proc. DAC*, June 2020, pp. 1–6.
[37] "CUGR," https://github.com/cuhk-eda/cu-gr.
[38] S. Dolgov, A. Volkov, L. Wang and B. Xu, "2019 CAD Contest: LEF/DEF based global routing," *Proc. ICCAD*, Nov. 2019, pp. 1–4.
[39] "TAU Contest 2020," https://sites.google.com/view/taucontest2020/home.
[40] A. B. Kahng, S. Kumar and T. Shah, "A no-human-in-the-loop methodology toward optimal utilization of EDA tools and flows," *DAC Work-in-Progress poster*, June 2018. https://vlsicad.ucsd.edu/MAB/
[41] "Rsyn," https://github.com/RsynTeam/rsyn-x.
[42] J. A. Roy, F. Koushanfar, I. L. Markov, "Ending piracy of integrated circuits," *IEEE Computer*, 43(10), pp. 30–38, Oct. 2010.
[43] J. Rajendran, M. Sam, O. Sinanoglu, and R. Karri, "Security analysis of integrated circuit camouflaging," *Proc. CCS*, Nov. 2013, pp. 709–720.
[44] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," *Proc. Host*, May 2015, pp. 137–143.
[45] M. Yasin, B. Mazumdar, J. J. V. Rajendran and O. Sinanoglu, "Sarlock: SAT attack resistant logic locking," *Proc. HOST*, May 2016, pp. 236–241.
[46] Y. Xie and A. Srivastava, "Anti-SAT: Mitigating SAT attack on logic locking," *IEEE TCAD*, 38(2), pp. 199–207, Feb. 2019.
[47] P. Chakraborty, J. Cruz, and S. Bhunia, "SAIL: Machine learning guided structural analysis attack on hardware obfuscation" in *Proc. AsianHOST*, Dec. 2018, pp. 56–61.
[48] "IWLS 2020 Programming Contest: ML+LS," https://iwls.org/.