DATC RDF-2019: Towards a Complete Academic Reference Design Flow

Invited Paper

Jianli Chen*, Iris Hui-Ru Jiang†, Jinwook Jung‡, Andrew B. Kahng§, Victor N. Kravets‡, Yih-Lang Li¶, Shih-Ting Lin¶, and Mingyu Woo§

*Fuzhou University, Fuzhou, China
†National Taiwan University, Taipei, Taiwan
‡IBM T. J. Watson Research Center, Yorktown Heights, NY, USA
§UC San Diego, La Jolla, CA, USA
¶National Chiao Tung University, Hsinchu, Taiwan

***Abstract*—We describe a new *RDF-2019* release of the IEEE CEDA DATC Robust Design Flow (RDF). RDF-2019 enhances the DATC RDF to span the entire RTL-to-GDS IC implementation flow, from logic synthesis to detailed routing. The new release represents a significant revision of the previously-reported RDF-2018 flow. Noteworthy *vertical* extensions include addition of logic synthesis starting from pure behavioral RTL Verilog RTL; floorplanning that includes initial DEF creation, I/O placement and PDN layout generation; and clock tree synthesis between placement legalization and global routing. A number of *horizontal* extensions to RDF are achieved by incorporating additional tool options at the static timing analysis, global placement, gate sizing, and detailed routing stages of the flow. Further, for the first time, multiple open-source realizations of the entire RDF tool chain are available. Last, RDF-2019 provides significantly enhanced support of and interoperability with industry-standard tools and design formats (LEF/DEF, SPEF, Liberty, SDC, etc.). We illustrate the configuration and use of RDF-2019, with example results on open as well as commercial design enablements.**

I. Introduction

Winning tools from academic research contests have for many years offered innovative solutions to modern design challenges. Building upon these outstanding point tools, the IEEE CEDA Design Automation Technical Committee (DATC) [1] has developed an open reference design flow, called *DATC Robust Design Flow* (RDF), to facilitate research on flow-scale methodology and cross-stage optimizations. As detailed in a series of papers beginning at ICCAD-2016 [2]–[5], the DATC RDF seeks to (i) provide an academic reference flow from logic synthesis to detailed routing based on existing contest results; (ii) construct a database for design benchmarks and point tool libraries; and (iii) connect academic research to industry practitioners and designs by using industry-standard design input/output formats.

While the past several years have seen increased engagement and expansion of the RDF flow, several opportunities for added research impact and industry engagement have remained out of reach. We note three key challenges that have been faced by RDF. (1) The chaining of academic contest-winning tools alone leaves significant gaps that preclude research on flow-scale or cross-stage optimizations. For example, the last-reported *RDF-2018* flow [4], [5], misses such functions as initialization of floorplan DEF, I/O placement, macro placement, clock tree synthesis, and generation of SPEF from placed or routed DEF. (2) RDF has maintained a policy of incorporating both open-source and non-open-source (distribution via binaries, or with usage restrictions) tools, to honor academic creators' freedoms and preferences. However, closed-source binaries cannot be updated, e.g., to handle a new hierarchy delimiter case or an industrial format extension. And, a tool with "no commercial use" typically cannot be opened by industry collaborators (see [6], [7] for useful "Open Source 101 / 102" background). (3) Crucially, a number of academic contests out of necessity use testcases that embody strong simplifications to data models and/or industry formats. Perpetuating these simplifications (e.g., translating a Bookshelf variant to LEF/DEF), along with a focus on comparisons with previous works using (old) contest benchmarks, has kept academic research in a kind of parallel universe of "translated", as opposed to "standard", industry formats and testcases. Tools developed in this parallel universe are often unable to accept real-world designs and enablements.

The latest *RDF-2019* release that we describe in this paper makes significant progress toward overcoming the above barriers. RDF-2019 leverages recent academic tool developments in the OpenROAD project [8], [9] to add previously-missing steps such as floorplanning, I/O placement, power planning and clock tree synthesis. These *vertical flow extensions* lead to a full academic flow from behavioral (RTL) Verilog to final detail-routed DEF. RDF-2019 also includes *horizontal flow extensions* that add numerous alternatives to existing tools. RDF-2019 now provides many paths from Verilog to routed DEF, including paths that are entirely open-source and capable of delivering DRC-clean layout in a commercial (65nm) foundry enablement.

II. RDF-2019 Flow Evolution

Previous accounts of the DATC RDF flow [2]–[5] document its origin as a composition of academic point-tool binaries for logic synthesis, placement, timing analysis, gate sizing, global routing, and detailed routing. These binaries are interfaced via transitional scripts that enable data exchange between tools of

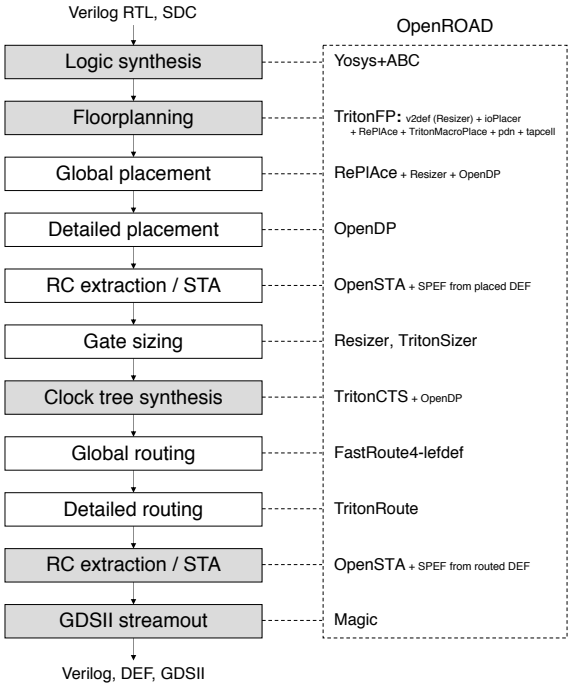

Fig. 1. Overview of RDF-2019 flow. It unifies RDF-2018, OpenROAD and additional tools. Vertical extensions made in this year are highlighted in gray.

other domains. The RDF flow continues to be maintained in this form, both on a Jenkins Pipeline-based server and in a commercial cloud deployment; see the README at [10].

As noted above, our current *RDF-2019* includes both horizontal and vertical extensions of the previous *RDF-2018* flow. Figure 1 shows the past year's advances that have led to the current state of RDF-2019. Vertical extensions from RDF-2018 to RDF-2019 include:

- Logic Synthesis from RTL (Yosys+ABC)
- Floorplan (TritonFP)
- Clock Tree Synthesis (TritonCTS)

Horizontal extensions from RDF-2018 to RDF-2019 include:

- Global Placement (FZUplace, RePlAce)
- Detailed Placement (OpenDP)
- Global Routing (FastRoute4-lefdef)
- Detailed Routing (Dr. CU, TritonRoute)
- Gate Sizing (Resizer, TritonSizer)
- Static Timing Analysis (OpenSTA)

Table I gives a concise summary of the recent RDF evolution. We give details of these extensions in the next section.

III. Details of RDF-2019 Flow Elements

A. Technology Libraries

RDF-2019 is tested on NanGate45 [11] and ASAP7 [12] technology libraries. With NanGate45, we have fully tested the RDF-2019 flow through the end of detailed routing. With ASAP7, the flow has been tested through the global routing

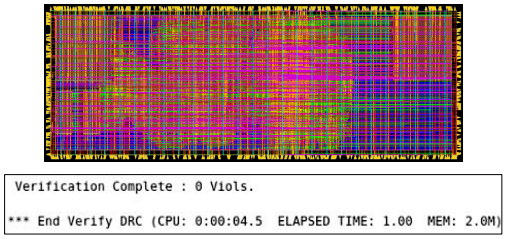

Fig. 2. A DRC-clean P&R result in TSMC 65LP obtained by the OpenROAD toolchain, which is available in RDF-2019.

stage.[1] In addition to those libraries, RDF-2019 also supports NCTUcell [13], a cell library based on the ASAP7 PDK which is generated in a fully automated way without any manual intervention. NCTUcell is newly included in the design flow this year and publicly available under RDF-2019. In addition to the cells in ASAP 7nm library, NCTUcell offers high-driving strength cells than ASAP7, including AND (x3–x16), BUF (x11–x14), Dlatch (x5–x8), INV (x11–x16), NAND (x3–x4), NOR (x3–x4), OR (x3–x18), XOR (x3–x16), and XNOR (x3–x16). The *RDF* library, based on ISPD-2012/2013 gate sizing contest cell library and reported in [4], has been found to have flaws in LEF and other elements. It is therefore no longer part of the supported technologies/libraries in RDF-2019.

Separately, tools in the OpenROAD project have demonstrated DRC-clean layout generation capability in the TSMC 65LP foundry node [14], [15]. They are available in RDF-2019 as noted above, and hence RDF-2019 can also support TSMC 65LP library if it is configured to use the OpenROAD toolchain (e.g., RePlAce, TritonCTS and TritonRoute). Figure 2 shows the DRC-clean P&R result in TSMC 65LP enablement of a design block (bsg_manycore_tile, "VanillaBean" [16]) with 17k standard-cell instances and four SRAM macros, obtained by running the entire RTL-to-GDS OpenROAD tool chain.

Note that only NanGate45 is, to the best of our knowledge, freely distributable. The ASAP7 PDK and other design enablement is not usable within commercial entities, and cannot be redistributed. This has detracted from its use at the interface between academic research and industry practice. Memory generators are a key gap in RDF-2019; as of this writing, we are not aware of any available memory generator for NanGate45. We are hopeful that memory generators from UC Santa Cruz (OpenRAM [17]), Yale (AMC [18]) or elsewhere will be contributed for broad use, including in our flow.

B. Logic Synthesis

The synthesis step of RDF-2019 accepts the behavioral circuit specification as Verilog RTL, and produces a library-mapped netlist implementation. Verilog timing assertions as

[1]ASAP7 has several LEF58 design rules, as it seeks to emulate production 7nm foundry rules. These LEF58 rules are not fully supported by publicly available academic detailed routers, which are typically based on the ISPD-2018/2019 contests.

TABLE I
VERTICAL AND HORIZONTAL EXTENSIONS MADE IN THE RDF-2019 FLOW

Component	RDF-2018 [4]	RDF-2019 Extension
Logic synthesis (RTL)	-	Yosys+ABC
Logic synthesis (gate-level)	ABC	-
Floorplanning	-	TritonFP
Global placement	NTUPlace3, ComPLx, mPL5/6, FastPlace3-GP, Capo, Eh?Placer	FZUplace, RePlAce
Detailed placement	FastPlace3-DP, MCHL-T	OpenDP
Clock tree synthesis	-	TritonCTS
Global routing	NCTUgr, FastRoute4.1, BFG-R	FastRoute4-lefdef
Detailed routing	NCTUdr	Dr. CU, TritonRoute
GDSII streamout	-	Magic
Gate sizing	USizer2012, USizer2013	Resizer, TritonSizer
Parasitic extraction	-	OpenROAD Utilities (PEX)
Timing analysis	OpenTimer, iTimerC	OpenSTA
Libraries/Technologies	ISPD-2012/2013 Contests, ASAP-7nm	NanGate45, NCTUcell

well as the standard SDC constraints format are supported in the flow. Up to RDF-2018, only structural (gate-level) Verilog was accepted as input to synthesis. In RDF-2019, the Yosys+ABC extension allows for more general RTL specifications, making the flow encompass RT-level Verilog as a starting point for implementation.

C. Floorplanning

Floorplanning in the RTL-to-GDS flow begins with logic synthesis outputs and ends with a DEF that is suitable for standard-cell global placement, i.e., with completed I/O placement, macro placement, PDN (P/G mesh) layout, and welltap/endcap placement. The floorplanning step of RDF-2019 is performed by *TritonFP*, whose first step uses [19] to generate an initial floorplan DEF from the post-synthesis structural Verilog netlist. The global placement tool *RePlAce* [20], [21] is an open-source electrostatics-based placer that supports mixed-size circuits and timing-driven placement. Its use in RDF-2019 includes seeding the *TritonMacroPlace* macro placement with a timing-driven mixed-sized placement solution. To leave a simpler problem for later placement and routing steps, *TritonMacroPlace* divides the layout region into four quadrants and uses a modified *ParquetFP* [22] to pack and snap macros into corners of the layout.

I/O placement is another step that is new in RDF-2019. The *ioPlacer* [23] code developed at UFRGS heuristically determines IO pin locations through applications of the Hungarian matching algorithm. Input to the I/O placement step consists of the initialized floorplan DEF and the post-synthesis gate-level netlist that can include macros. The ioPlacer code also accepts constraints on pin layers and allowed/disallowed segments of the region boundary along which I/Os are placed. Last, TritonFP also includes codes and scripts for PDN (P/G mesh) layout and tapcell insertion.

D. Global Placement

In RDF-2019, two recent global placers FZUplace [24] and RePlAce [20], [21] are added to the six placers already present in RDF-2018. FZUplace provides an analytical solution to the equation to calculate the potential energy of an electrostatic system. A fast computation scheme of Poisson's equation yields an effective and efficient global placement algorithm. RePlAce proposes a new density function that comprehends local overflow of area resources; this enables a constraints-oriented local smoothing at per-bin granularity. Extensions are also given to address timing and routability. In timing-driven mode, RePlAce optimizes both WNS and TNS criteria, guided by OpenSTA [25] and iterative reweighting of timing-critical nets using the method of [26].

E. Detailed Placement

An open-source detailed placer OpenDP has been newly added in RDF-2019. Developed for the ICCAD-2017 Mixed-Cell-Height Standard Cell Legalization Contest, OpenDP [27] performs mixed-height legalization based on fence region constraints. It first performs pre-legalization and mixed-height standard cell legalization, then improves solution quality using simulated annealing to reduce displacement from the original cell locations.

F. Clock Tree Synthesis

Clock tree synthesis (CTS) is another vertical extension seen in RDF-2019. *TritonCTS* [28] is an open-source extension of the academic code reported in [29], with usage of mathematical programming solvers replaced by heuristic codes that are more amenable to open-sourcing. Inputs to the CTS step consist of a gate-level netlist (.v) and a placed DEF; standard-cell library information is also needed for a lookup-table characterization step that informs slew- and delay-constrained buffering. TritonCTS calls OpenDP internally to legalize clock buffers inserted during CTS.

G. Global Routing

A new open-source global router, FastRoute4-lefdef [30], has been added to RDF-2019. FastRoute4-lefdef is based on the well-known open-source FastRoute4.1 [31] from Iowa State University, and is written on top of Rsyn [32], the open-source physical design framework developed by the Federal University of Rio Grande do Sul (UFRGS). It "natively" supports LEF/DEF format. Importantly, it also outputs the

"route guide" format established by the ISPD-2018 and ISPD-2019 Initial Detailed Routing Contests, to achieve a well-defined handoff between global and detailed routing in the academic research world. (Such a handoff is always hidden inside any commercial place-and-route tool.) We note that previous academic global routers are largely unable to "natively" read LEF/DEF input files, as they were developed for the formats of the ISPD-2007 and ISPD-2008 contests (.gr), or of the ISPD-2011, DAC-2012 and ICCAD-2012 contests (.route). Thus, RDF-2019 provides a preprocessor utility that creates global routing benchmark inputs in the ISPD-2007/2008 contest format from LEF/DEF placement instances (recall the challenge of "translated" versus "standard" industry formats noted above).

H. Detailed Routing

In RDF, the detailed routing step takes as input industry-standard LEF/DEF along with the *route guide* (.guide) format established by the ISPD-2018 Initial Detailed Routing Contest [33]. Similar to the case of global routing, the RDF flow provides a utility to ensure that the detailed routing inputs are well-formed. The utility checks if (i) the track information in given DEF file conforms with the metal pitch information defined in LEF, and (ii) there are any missing design rules, e.g., minimum area and/or end-of-line spacing on layers in LEF. The utility also serves as a .route guide checker, to examine whether the global router ignores the routes of the overflowed nets. Since existence of a global route guide for each net is compulsory for some detailed routers,[2] if a net without a global guide is found, the checker will add for this net a global routing guide enclosed by the bounding box of all of the net's pins.

The ISPD-2019 contest-winning team's detailed router, Dr. CU [34], has been added to the RDF-2019 flow. With the aid of global route guides, Dr. CU has excellent performance in terms of quality and runtime. However, if there are some nets with no initial global guide from the global router, Dr. CU becomes slow because the range of the global guide offered by the checker may be very large as compared to a global guide identified by the global router.

TritonRoute [35] is also added in RDF-2019. With its integrated DRC engine, TritonRoute is the first open-source academic detailed router which is capable of delivering DRC-clean routing solution in a commercial foundry node (e.g., TSMC 65LP, with Arm standard cells and generated memories) with a restricted selection of standard cells and macros.

I. GDSII Streamout

Magic VLSI [36] is a VLSI layout tool originally written in the 1980s at UC Berkeley. Now maintained by developer Tim Edwards, it supports various foundry process design kits (PDKs). It also provides a DEF-to-GDSII flow which has been demonstrated to successfully convert routed DEF into GDSII (with correctness confirmed using commercial physical verification tools), in both commercial foundry 65LP and open NanGate45 technologies.

[2]In ISPD-2018/2019 Initial Detailed Routing Contest benchmark circuits, every net has associated routing guides. Hence, academic detailed routers based on ISPD-2018/2019 contests may assume that a routing guide is available for every net.

```
- stage: synth
  tool: yosys-abc
  parms: { max_fanout: 16, script: resyn2rs }

- stage: floorplan
  tool: TritonFP
  parms: { utilization: 0.5, aspect_ratio: 1 }

- stage: global_place
  tool: RePlAce
  parms: { target_density: 0.8, timing-driven: true }

- stage: detail_place
  tool: opendp

- stage: cts
  tool: TritonCTS
  parms: { target_skew: 50, max_fanout: 32 }

- stage: global_route
  tool: FastRoute4-lefdef

- stage: detail_route
  tool: TritonRoute
  parms: { time_out: 36000 }

- timer: OpenSTA
- sizer: TritonSizer
```

Fig. 3. RDF-2019 flow configuration example.

J. Timing Analysis

OpenSTA [25] has been developed over nearly 20 years and is an open-sourced version of the commercial Parallax timer. OpenSTA is publicly available on GitHub [25] and supports other timing-aware tools in RDF-2019 such as RePlAce [21], Resizer [19], TritonCTS [28] and TritonSizer [37]. A crucial benefit of OpenSTA is its production-proven (as an engine embedded in over 15 EDA companies' products) handling of SDC constraints as well as Verilog, cell libraries and parasitic files seen with commercial designs.[3]

K. Gate Sizing

Two gate sizers, Resizer and TritonSizer, are added in RDF-2019. Resizer [19] performs buffering of long nets and upsizing of gates for placed designs. Resizer supports all OpenSTA commands, and can take LEF/DEF format as inputs and outputs. TritonSizer [37] is a gate sizer that optimizes leakage and dynamic power, and fixes timing violations. It is the result of several years' evolution from an original ISPD-2013 Discrete Gate Sizing contest metaheuristic. TritonSizer has achieved strong leakage reduction results on a commercial mixed-signal SoC product (with multi-corner signoff at 35 timing views) [39] and supports integration with leading commercial STA engines as well as OpenSTA.

[3]The RDF-2019 flow also adds capability to generate SPEF files from DEF layout, closing the loop to timing-driven layout capability. SPEF generation from placed DEF (using FLUTE to estimate Steiner trees), and from routed DEF, is performed by utilities at [38].

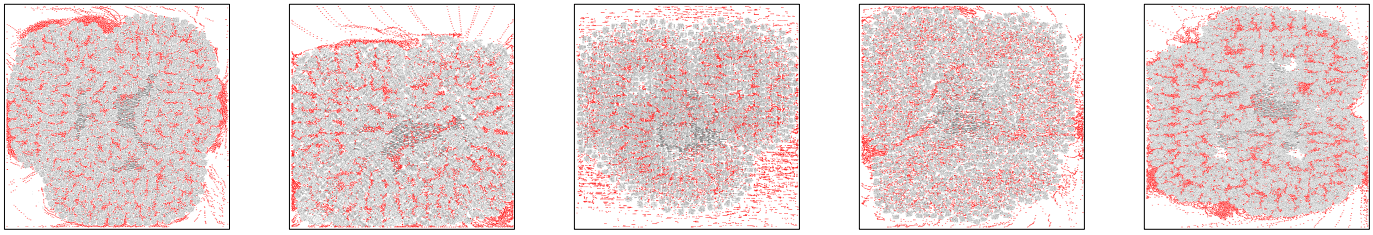

Fig. 4. Placement results of des3_perf from IWLS 2005 benchmarks on NanGate45: (a) RePlAce, (b) ComPLx, (c) NTUplace3, (d) Eh?Placer, and (e) FZUplace. OpenDP was used to perform detailed placement. Flip-flops are highlighted in red.

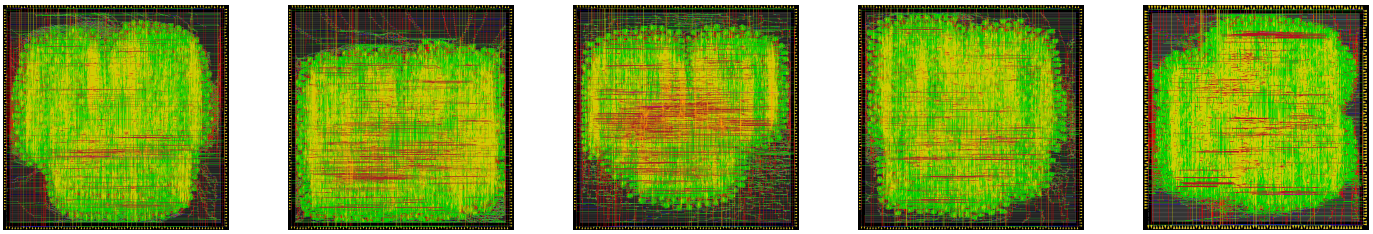

Fig. 5. Detailed routing results obtained using FastRoute4-lefdef and TritonRoute for the placements (a)–(e) in Fig. 4.

L. Flow Configuration

In RDF-2019, a design flow is composed using a set of configuration files in YAML format. In a design configuration file, a user specifies the top module name, an SDC file, Verilog RTL source files, and the library to be used throughout the flow. Flow configuration includes the point-tool name of each stage, along with parameters such as target density and maximum fanout. There is also a library configuration file, which specifies the paths to timing libraries, technology and cell LEF files, and other PDK-related parameters and setups. Figure 3 shows an example flow configuration file (see [10] for more detail on the flow configuration).

IV. Demonstration

To illustrate the capability of RDF-2019, we show results of a small experiment that juxtaposes DRC violations after detailed routing, versus placement half-perimeter wirelength (HPWL). For a circuit benchmark des3_perf, a behavioral Verilog RTL from IWLS 2005 benchmarks, we run the entire flow from logic synthesis through detailed routing on NanGate45. We use five global placers available in RDF-2019 and obtain five different placement solutions; detailed placement is done using OpenDP (see Fig. 4). Detailed routing results for each of the placements are obtained using FastRoute4-lefdef and TritonRoute, as shown in Fig. 5.

Table II shows the results.[4] The second and third columns of the table show the DRC violation counts and the routed wirelengths (RtWLs), and the fourth column gives the placement HPWLs. It can be seen that the traditional measure of placement quality (i.e., HPWL), and even final RtWL, shows miscorrelation with actual DRC violation counts. These results highlight the need for future research in flow-scale contexts, or addressing cross-stage optimization rather than any single flow stage.

TABLE II
Detailed routing results and placement HPWLs for various global placement results

Placer	DRC Viol.	RtWL (mm)	HPWL (mm)
GP1	40	2.71	2.23
GP2	386	2.73	2.27
GP3	20	2.84	2.36
GP4	28	2.85	2.37
GP5	38	3.04	2.56

We also demonstrate support for the ASAP7 library in RDF-2019. As noted above, RDF-2019 can support ASAP7 from logic synthesis through global routing. Figure 6 (a) shows a placement result of des3_perf obtained by RePlAce and OpenDP. We perform global routing on the placement using NCTUgr; the resulting global routing congestion map is shown in Fig. 6(b) and (c).

V. Conclusion

We have presented RDF-2019, an updated version of the IEEE CEDA DATC's Robust Design Flow. With its numerous horizontal and vertical extensions, RDF-2019 brings academic researchers and industry practitioners closer together, and serves as a unifying framework for academic research works

[4] As the purpose of this demonstration is not to benchmark different global placers, we sort the results in ascending order of HPWL and hide the names of the placers by assigning arbitrary names to them (e.g., GP1).

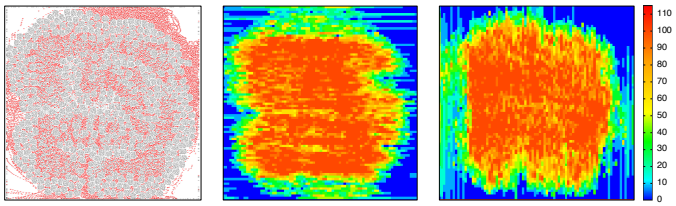

Fig. 6. Placement and global routing results for des3_perf using the ASAP7 library: (a) placement, and routing congestion maps of (b) M5 and (c) M6 layers. RePlAce and OpenDP were used to generate the placement, and NCTUgr was used to generate the global routing.

in the RTL-to-GDS tools space. Today, RDF-2019 embraces two flavors of academic tool flows: (i) flows with genealogy in academic contests that use "interpreted" industry formats (e.g., subsets of LEF/DEF recast in Bookshelf format variants), and (ii) flows that attempt to support "standard" industry formats. Importantly, academic tools in RDF can be either closed source or open source, staying true to the original RDF ethos that researchers must always be free to choose how their code is made available to, or used by, others.[5]

As of now, there are only three intersections where tools can be freely swapped: Logic Synthesis, Static Timing Analysis, and Detailed Routing. In future RDF-2020+ versions of our framework, we hope to see more academic tools entering an RDF world that increasingly brings researchers and practitioners closer together, and enables more efficient and exciting research progress.

Acknowledgment

This work was supported by IEEE CEDA Design Automation Technical Committee (DATC). We thank the OpenROAD project team for many contributions and support for RDF-2019. We also would like to thank Myung-Chul Kim of IBM and Igor Markov of of University of Michigan for their generosity in providing the ComPLx placer binary. Also, we would like to thank Shinichi Nishizawa of Fukuoka University and Hidetoshi Onodera of Kyoto University for supporting NCTUcell release under RDF-2019.

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," in *Proc. Int. Conf. Comput.-Aided Design (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," in *Proc. Int. Conf. Comput.-Aided Design (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," in *Proc. Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2018, pp. 37:1–37:4.

[5]Thus, RDF-2019 is a proper superset of the OpenROAD project. This will always be the case since OpenROAD by its design and policy only includes open-source, while RDF includes both open and closed source.

[5] ——, "DATC RDF: An open design flow from logic synthesis to detailed routing," in *Proc. Workshop Open-Source EDA Tech. (WOSET)*, Nov. 2018, pp. 6:1–6:4.
[6] "Open Source 101," https://j.mp/eri19-foss101.
[7] "Open Source 102," https://j.mp/eri19-foss102.
[8] T. Ajayi, V. A. Chhabria, M. Fogaça, 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," in *Proc. Design Autom. Conf. (DAC)*, Jun. 2019, pp. 76:1–76:4.
[9] "The OpenROAD Project," https://github.com/The-OpenROAD-Project.
[10] "RDF-2019 GitHub," https://github.com/ieee-ceda-datc/RDF-2019.
[11] "NanGate FreePDK45 Generic Open Cell Library," http://projects.si2.org/openeda.si2.org/projects/nangatelib.
[12] 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.*, vol. 53, pp. 105–115, Jul. 2016.
[13] 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," in *Proc. Design Autom. Conf. (DAC)*, Jun. 2019, pp. 1–6.
[14] ERI Summit 2019, "OpenROAD: Foundations and realization of open, accessible design," https://theopenroadproject.org/openroad_event/openroad-presentation-at-the-eri-summit-2019.
[15] "OpenROAD Alpha Release Github page," https://github.com/The-OpenROAD-Project/alpha-release.
[16] https://bitbucket.org/taylor-bsg/bsg_manycore/src/vanilla_bean/.
[17] "OpenRAM," https://openram.soe.ucsc.edu/.
[18] S. Ataei and R. Manohar, "AMC: An asynchronous memory compiler," in *Proc. Int. Symp. Async. Circuits Syst. (ASYNC)*, May 2019, pp. 1–8.
[19] "Resizer," https://github.com/The-OpenROAD-Project/Resizer.
[20] C.-K. Cheng, A. B. Kahng, I. Kang, and L. Wang, "RePlAce: Advancing solution quality and routability validation in global placement," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, pp. 1–14, Early Access, 2019.
[21] "RePlAce," https://github.com/The-OpenROAD-Project/RePlAce/.
[22] "ParquetFP," http://vlsicad.eecs.umich.edu/BK/parquet.
[23] "ioPlacer," https://github.com/The-OpenROAD-Project/ioPlacer.
[24] W. Zhu, Z. Huang, J. Chen, and Y.-W. Chang, "Analytical solution of Poisson's equation and its application to VLSI global placement," in *Proc. Int. Conf. Comput.-Aided Design (ICCAD)*, 2018, pp. 2:1–2:8.
[25] "OpenSTA," https://github.com/The-OpenROAD-Project/OpenSTA.
[26] M. Fogaça, G. Flach, J. Monteiro, M. Johann, and R. Reis, "Quadratic timing objectives for incremental timing-driven placement optimization," in *Proc. Int. Conf. Electron. Circuits Syst. (ICECS)*, 2016, pp. 620–623.
[27] "OpenDP," https://github.com/The-OpenROAD-Project/OpenDP.
[28] "TritonCTS," https://github.com/The-OpenROAD-Project/TritonCTS.
[29] K. Han, A. B. Kahng, and J. Li, "Optimal generalized H-Tree topology and buffering for high-performance and low-power clock distribution," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, pp. 1–14, Early Access, 2018.
[30] "FastRoute4-lefdef," https://github.com/The-OpenROAD-Project/FastRoute4-lefdef.
[31] Y. Z. Yue Xu and C. Chu, "FastRoute 4.0: Global router with efficient via minimization," in *Proc. Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2009.
[32] "Rsyn," https://github.com/RsynTeam/rsyn-x/.
[33] S. Mantik, G. Posser, W.-K. Chow, Y. Ding, and W.-H. Liu, "ISPD 2018 initial detailed routing contest and benchmarks," in *Proc. Int. Symp. Phys. Design (ISPD)*, Mar. 2018, pp. 140–143.
[34] "Dr.CU," https://github.com/cuhk-eda/dr-cu.
[35] "TritonRoute," http://github.com/The-OpenROAD-Project/TritonRoute.
[36] "Magic," https://github.com/The-OpenROAD-Project/magic.
[37] "TritonSizer," https://github.com/The-OpenROAD-Project/TritonSizer.
[38] "Parasitic-Extraction–OpenROAD Utilities," https://github.com/The-OpenROAD-Project/OpenROAD-Utilities/tree/master/PEX.
[39] H. Fatemi, A. B. Kahng, H. Lee, J. Li, and J. P. de Gyvez, "Enhancing sensitivity-based power reduction for an industry IC design context," *Integration, the VLSI Journal*, vol. 66, pp. 96–111, 2019.