This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MDAT.2022.3179247, IEEE Design and Test

RosettaStone: Connecting the Past, Present and Future of Physical Design Research

†‡Andrew B. Kahng, ‡Minsoo Kim, †Seungwon Kim and ‡Mingyu Woo
†CSE and ‡ECE Departments, UC San Diego, La Jolla, CA, USA
{abk, mik226, sek006, mwoo}@ucsd.edu

***Abstract*—Productivity of academic research on physical design, as well as translation of research results into industry practice, is hampered by lack of a standard backplane for integration and testing of new methods. Ad-hoc scripting and file-based communication have been used to stitch together academic tools, but such approaches are cumbersome and brittle. We describe *RosettaStone*, an open and extensible foundation which leverages a standard physical design data model (LEF/DEF 5.8) and open-source database implementation (OpenDB [12]) to effectively connect the academic physical design field's past, present and future. RosettaStone's shared data model enables richer integrations, flow contexts, and assessments for research. We demonstrate the use of OpenDB to make inter-stage data transformations that improve overall flow outcomes. We also show how RosettaStone enables integration of closed-source research tools and non-standard data formats for robust evaluation with modern technologies and testcases. RosettaStone has recently been included in the 2021 release of the IEEE CEDA DATC Robust Design Flow (RDF [7]).**

***Index Terms*—Physical design, open-source, standard physical design data model, academic tool, Bookshelf, OpenROAD, OpenDB**

I. Introduction

Academic research in the physical design ("RTL-to-GDS") domain has been largely pursued in separate subfields such as global placement, detailed placement, clock tree synthesis, global routing, detailed routing, etc. Each subfield has a rich literature that includes works developed for influential academic contests. However, standalone optimizations are not properly evaluated without adequate flow context. For example, minimizing an objective function such as "total half-perimeter wirelength" does not lead to a routable standard-cell placement. Thus, a standard backplane is required to support academic EDA research and translation of research results into industry practice. To this end, the field has pursued two main approaches: (i) aggregation and chaining of academic research codes, and (ii) mappings between contest benchmarks and a standard data model to make research codes more accessible.

The first approach is embodied in the IEEE CEDA DATC's Robust Design Flow (RDF [7]), which integrates prominent academic tools into RTL-to-GDS tool chains. However, connecting the steps of such tool chains is challenged by lack of an underlying physical design data model (e.g., LEF/DEF or OpenAccess) that is somehow consistent with the various simplifications and obfuscations that are needed when particular contests are conceived. The three most recent RDF releases [6] [7] partially address this by incorporating OpenROAD [1], an RTL-to-GDS tool that supports a modern back-end data model and standard formats. However, closed-source academic tools in the RDF, notably those written for contests, are "frozen in time": they exist in a parallel universe of non-standard data models, formats and testcases. Such tools are typically unable to accept real-world designs and enablements. Up to now, contest format readers/writers and file-based wrapper scripts have been the thread that connects past academic tools into the RDF.

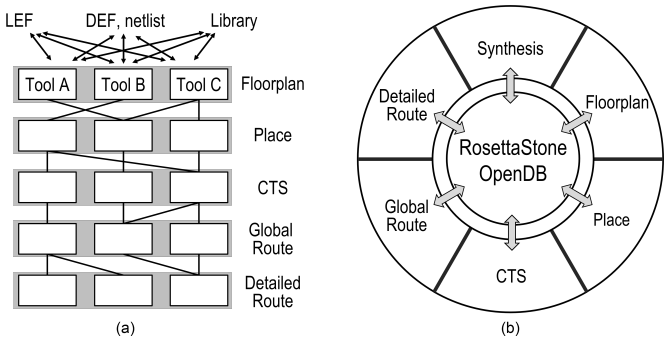


Fig. 1: RosettaStone for physical design. (a) Before: vertical chaining of tools with file-based communication. (b) Today: tool integration with industry-standard data model and high-quality OpenDB implementation via RosettaStone.

The second approach is exemplified by the A2A work of [8]. A2A focuses on the integration of benchmarks, in particular, mapping academic contest benchmark formats into a standard (LEF/DEF) data model. This enables contest benchmarks to be run in commercial tools for "apples-to-apples" assessments of both commercial and academic tools. (By contrast, RDF focuses on vertical linkage of flow stages, and on academic tools, benchmarks and evaluation metrics.) Unfortunately, A2A's file-based interfaces and reliance on closed-source commercial P&R tool scripting make it difficult to update.

In this paper, we present *RosettaStone*, a permissively open-sourced project that enables past academic tools and benchmark suites to be integrated into a complete, modern RTL-to-GDS foundation for physical design research. Notably, all *Bookshelf* [2] based contest formats and benchmark suites since 2005 are now connected (along with LEF/DEF data) via RosettaStone. Figure 1 shows the contrast between (a) the previous partial connectivity of file-based tool chains, and (b) the flexible tool integration that RosettaStone enables via consistent data model and database, along with robust format conversions.

RosettaStone comprises a set of Python and Tcl scripts built on top of OpenDB. We leverage the OpenDB Python API to maximize user convenience and ease of maintenance. Source code is available in GitHub (https://github.com/ABKGroup/RosettaStone) with more than 30 example benchmarks. RosettaStone in theory works with any technology, and we have

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MDAT.2022.3179247, IEEE Design and Test

validated RosettaStone using four open-source technologies (ASAP7, NanGate45 and SkyWater130HS/HD)[1] as well as a commercial foundry technology ($12nm$) with cell libraries from a leading IP provider.

Figure 2 illustrates the scope of RosettaStone and how it completes the vision of previous efforts such as RDF and A2A. RosettaStone goes beyond tool chaining, and enables deeper integrations whereby (i) flow stages can consider or co-operate with subsequent stages, or (ii) academic tools can be cross-evaluated on commercial/academic benchmarks. RosettaStone not only connects past academic tools to the present and future of physical design research, but also enables future academic contests to be framed in a complete flow context with canonical evaluations such as post-route timing or number of DRC violations. Below, our case studies illustrate the connection of past contest formulations and formats to newer academic and commercial technologies, and how additional physical design considerations (e.g., standard-cell padding during placement) can be incorporated into testcases or flows.

Our main contributions are described as follows.

- We develop *RosettaStone* with an industry-standard LEF/DEF 5.8 data model and open-source database (OpenDB).
- We describe challenges and solutions seen in mapping past academic benchmarks to an industry-standard data model, and we achieve new benchmark integrations of ISPD-2005, ISPD-2006, ISPD-2011 and DAC-2012 academic benchmarks. All Bookshelf-based contests since 2005 are now unified with the LEF/DEF data model via RosettaStone.
- We connect past academic contest frameworks and contest-winning academic tools to current industry technologies and testcases, leveraging OpenDB Python APIs.
- We enable two additional physical design features, cell padding and layer capacity adjustments, to show the extensibility and flexibility of RosettaStone. Inter-tool communication with these features is a key to pursue further design optimization in the physical design flow.
- We present case studies to demonstrate (i) integration of past tools and benchmarks with current tools and design enablements, and (ii) use of RosettaStone to extend past contest frameworks as well as inter-tool communication for improved flow outcomes.

II. OpenDB: Open-Source Physical Design Data Model

As observed by [12], a 1980s style file-based EDA flow does not work very well for inter-stage communications. OpenAccess is a well-known physical design data model which provides plentiful APIs for physical design, but is not open-sourced and cannot freely cross academia-industry boundaries. OpenDB is a physical design data model with concrete, industry-strength and open-source realization. The source code[2] of OpenDB has been released since 2019 as the foundation data model and database implementation for the OpenROAD project [1]. (Over 670 commits to OpenDB have been made since its open-sourcing.) Due to these reasons, we use OpenDB as a foundation of RosettaStone for boosting of physical design research. In this section, we describe the basic structure and key features of OpenDB.

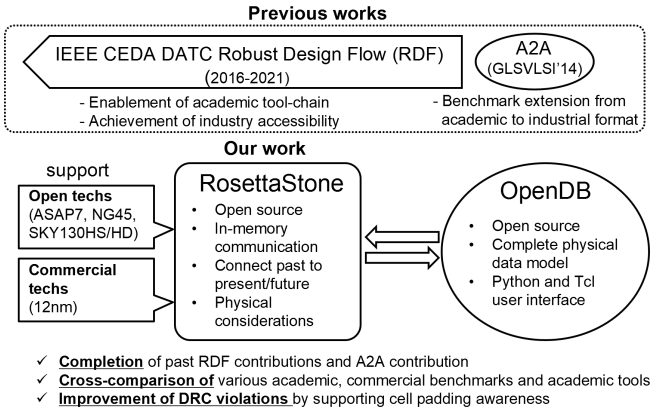


Fig. 2: RosettaStone completes the vision of previous research enablements (RDF [7] and A2A [8]). We demonstrate new cross-comparisons as well as extensions to improve flow outcomes.

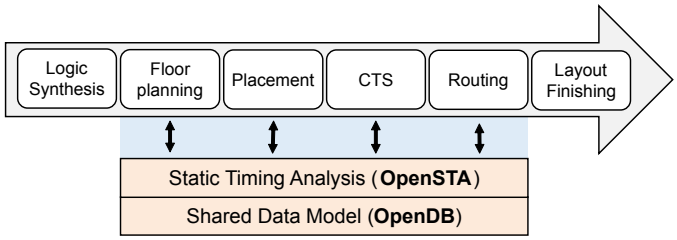


Fig. 3: SP&R flow of OpenROAD [1] and physical design (floorplanning through detailed routing) scope of RosettaStone integrations. Memory-based communication is enabled by OpenDB. Figure adapted from [9] [12].

A. OpenDB Overview

OpenDB has been implemented in the OpenROAD project [1] to enable incremental optimizations with tight coupling between flow stages. Figure 3 shows how OpenDB and OpenSTA together realize the central physical design data model (i.e., netlist, layout and timing) in the OpenROAD flow. OpenDB provides an incremental architecture to realize the full LEF/DEF 5.8 physical design data model, with support of physical hierarchy. In OpenDB, a *shared netlist adapter* enables netlist updates, such as the addition and deletion of logical components in the design. A *shared physical or data model adapter* enables updates of physical information, such as locations/orientations of placed instances and locations/layers of routed nets. The physical data model allows tools from different flow stages to work together in design optimization. For example, when the location and size of a placed instance is changed, the router can rip-up existing routed nets and reroute them; the timer can update slacks with new delays obtained with new estimated parasitics. More details of OpenDB are given in [12].

Main classes in OpenDB map to main keywords of LEF/DEF, e.g., OpenDB's *dbMaster*, *dbInst*, *dbNet* and *dbBTerm* correspond to LEF/DEF's *MACRO*, *COMPONENTS*, *NETS* and *PINS* keywords. In addition, OpenDB is tightly coupled with OpenSTA[3] for incremental static timing analysis.

[1]Respective URLs: http://asap.asu.edu/asap, https://si2.org/open-cell-library and https://github.com/google/skywater-pdk.

[2]https://github.com/The-OpenROAD-Project/OpenROAD/tree/master/src/odb

[3]https://github.com/The-OpenROAD-Project/OpenSTA

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MDAT.2022.3179247, IEEE Design and Test

B. User Interfaces

There are many academic readers and writers for physical design data formats that are of low quality (that is to say, buggy, and with only partial support of LEF/DEF syntax and versions). Creating a single (and robust) reader/writer for industry-standard formats for physical design brings a significant benefit as a foundation for academic research. OpenDB supports LEF/DEF 5.8 and provides industrial-strength readers and writers for other physical design-related data formats (.sdc constraints, .spef parasitics, .lib timing/power models, etc.). Also, OpenDB can store and load design data in binary format by using the OpenDB Tcl commands, *save_db* and *load_db*.

In addition to C++ APIs, OpenDB also supports scripts based on high-level programming languages (Tcl and Python) by using the Simplified Wrapper and Interface Generator (SWIG). This brings an additional benefit for users in that they can easily build their own physical design methodology.

III. RosettaStone for Physical Design Research

Connecting the past, present and future academic physical design research presents a range of requirements. For example, *RePlAce* [3] and *FastRoute* [11] come from the academic contest "universe", but have been updated to handle the LEF/DEF data model because they are distributed as open source. On the other hand, *NTUplace3* [5] or *NCTU-GR* [10], while being landmark works in the history of physical design and contests, are not open-sourced and are thus "frozen in time": they require RosettaStone in order to robustly interface to modern LEF/DEF 5.8 testcases and flow. We have developed RosettaStone to (i) make academic benchmarks available with the industry-standard data model; (ii) support academic tools via robust, bidirectional Bookshelf-OpenDB conversion; and (iii) enable additional physical design features in the flow. As described below, RosettaStone goes beyond simple file conversion in ways that include the following.

- We open-source RosettaStone scripts to integrate multiple Bookshelf-based contest benchmarks.
- Benchmark integration supports any modern technology, and we have performed validations for four open-source and one closed-source (12nm) technology.
- Creation of fake macro cell LEF files with on-grid pins enables academic detailed routers to address benchmarks with macro cells.
- We use OpenSTA to break combinational logic loops caused by cell mapping, to unblock physical design flows. In A2A [8], this required use of commercial tools and closed-source scripting.
- We resolve physical information mismatches between Bookshelf and OpenDB by upscaling and downscaling.
- Academic tool limitations such as implicit assumptions and hardcoded parameters have been worked around, enabling robust tool integrations.
- Two additional physical design features, cell padding and layer capacity adjustments, are demonstrated to show the extensibility and flexibility of RosettaStone.

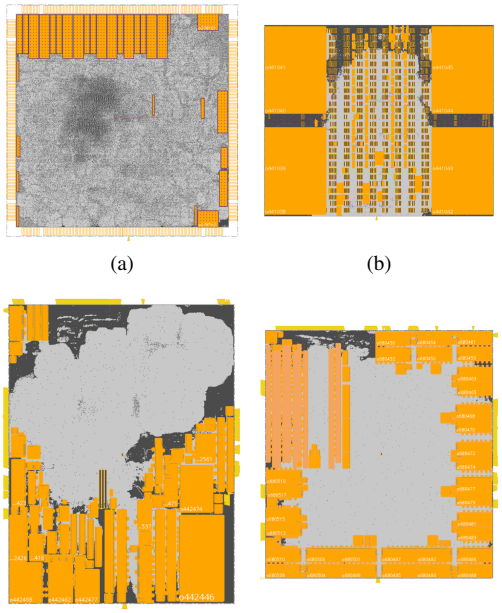


Fig. 4: Contest benchmarks converted from Bookshelf by RosettaStone. (a) NanGate45-bigblue1 (ISPD-2005), (b) SkyWater130HD-newblue2 (ISPD-2006), (c) NanGate45-superblue18 (ISPD-2011), and (d) ASAP7-superblue16 (DAC-2012). The standard cells are placed by a commercial P&R tool.

A. Academic Benchmark Integration with an Industry-Standard Data Model

Academic contest benchmarks do not have a complete set of required inputs for physical design. To overcome this limitation, A2A [8] proposes a benchmark integration flow with standard data format for both sizing and placement benchmarks, using ISPD-2011 benchmarks with ISPD-2012/2013 academic technology. RosettaStone extends this scope to include ISPD-2005, ISPD-2006, ISPD-2011 and DAC-2012 academic benchmarks.[4] Also, while A2A includes scripts for closed-source commercial tools, RosettaStone is completely implemented using OpenDB Python APIs, which enables its open-sourcing. Our benchmark integration overcomes five main challenges: (i) technology mapping, (ii) fake LEF generation, (iii) handling of combinational logic loops, (iv) unrealistic timing paths and (v) different Bookshelf formats.

First, RosettaStone must support the integration of academic benchmarks with modern P&R in any technology. The challenge here is that academic benchmarks may erase functional information for cells in the netlist. Thus, mapping must be performed to enable consumption by P&R tools. To do this, we first classify cells in a netlist as combinational or sequential. When cells have (i) one input pin, (ii) one output pin and (iii) width larger than $minSeqWidth$ placement sites, the cells are assumed to be sequential cells.[5] To determine $minSeqWidth$, we find all cells which have a single input and a single output and sort them by width. Then, $minSeqWidth$ is defined as

[4]In A2A, the benchmark integration flow has two directions: sizing and placement. In RosettaStone, we focus on enabling placement contest benchmarks as commercial formats. The gate sizing contest benchmarks are already based on commercial formats (Verilog, SDC and SPEF).

[5]We assume that sequential cells (flip-flops) only have a single output (Q or $\bar{Q}$) because academic technologies might not have multiple output pins (e.g., ASAP7). For sequential cells with two output pins (both Q and $\bar{Q}$), we connect only one output pin (Q) and leave the other pin ($\bar{Q}$) open.

the smallest width when at least 9% of cells have a larger width than $minSeqWidth$. All other cells are assumed to be combinational cells.[6] For each distinct width seen in a benchmark's combinational cells, we choose cells with the same or the closest width from the provided standard-cell library. If multiple candidates are available (i.e., there are multiple cells having the same (or, closest) width in the provided standard-cell library), we randomly assign the cell to one of the candidates.

Second, RosettaStone supports academic routers for benchmarks with macros and I/O ports. We create fake LEF files which contain macro cells. When creating macro LEF files, we further define macro obstructions (OBS) and I/O ports as described in contest benchmarks. Importantly, RosettaStone's benchmark conversion ensures that all converted macros and I/O ports are placed to honor manufacturing grids and design rules.

Third, since the combinational cells are randomly mapped, the physical design flow might be blocked by combinational logic loops. We break all the loops, creating new sequential cells and nets as necessary, using OpenSTA APIs.

Fourth, lengths of critical timing paths can be unrealistic due to random mapping of logic gates. To address this issue, we provide tunable logic path cutting. We call OpenSTA APIs to retrieve the worst timing paths and change 1-output/1-input combinational logic gates into flip-flops, or insert flip-flops and create associated output nets, so as to not exceed a user-defined maximum path length ($maxPathLength$).

For example, compared to ISPD-2006 benchmarks, ISPD-2011 benchmarks have an additional *terminal_NI* attribute for fixed I/O ports. The *terminal_NI* attribute allows overlaps of I/O ports on different layers. Therefore, academic tools created for ISPD-2006 benchmarks cannot parse the ISPD-2011 benchmarks. RosettaStone can select whether the *terminal_NI* attribute is used; when it is used, I/O ports are classified as *terminal_NI*. Figure 4 shows converted academic benchmarks and their placements by a commercial tool.

B. Robust Conversion for Academic Tools

Since we use OpenDB as the central database in RosettaStone, we must have robust bidirectional conversion between Bookshelf and OpenDB for better academic tool support. We achieve this by solving five problems: (i) infeasibility of off-grid routing; (ii) resolution mismatches; (iii) implicit assumptions and hardcoded parameters that create silent limitations in academic tools; (iv) defining appropriate tile (gcell) sizes for global and detailed routing; and (v) handling *fragmented rows*.

The first problem is that some academic detailed routers only support on-grid routing. In RosettaStone, during the fake LEF generation flow, we define grids having 2 $\times$ M3 pitch and 2 $\times$ M4 pitch for vertical and horizontal grids, respectively. The macro pins are aligned to these grids so that detailed routing can be accomplished with academic routers. The macro pin snapping makes the designs routable with the academic detailed routers.

The second problem is mismatches between Bookshelf and OpenDB. We find that academic tools can have overflows when we integrate contest benchmarks to new technologies. The enlarged values of coordinates of die and/or nodes cause an overflow error in academic tools that only support 'int' data type, whereas OpenDB and LEF/DEF support 'float' data type. To use such academic tools, we cannot simply scale down by dividing values by specific integers, due to the remainder of site width and height. Therefore, we scale down by dividing the coordinates by the greatest common divisor of site width and height. Scaling by using the greatest common divisor avoids distortion of the aspect ratios of die area and core area for integer type values. Due to the mismatches between Bookshelf and OpenDB, OpenDB must retain the original information to recover from any corruption induced by upscaling and downscaling.

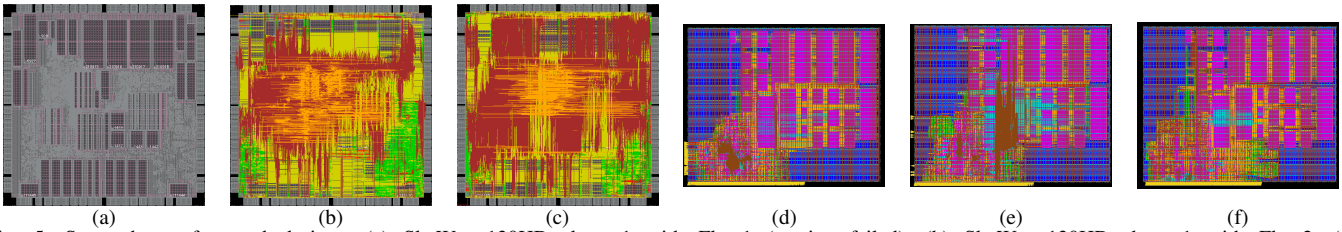
The third problem is that academic tools have their own assumptions and/or hardcoded parameters to obtain proper outputs based on contest requirements. For example, Bookshelf has three node types: movable nodes, fixed nodes (*terminal*) and fixed nodes with overlap (*terminal_NI*). The academic tools only support movable nodes and *terminal* nodes with the prefix 'o', *terminal_NI* nodes with the prefix 'p', and nets with the prefix 'n'. To address these limitations, we change instance and net names to have proper naming conventions for the tools. We proceed to inverse-mapping the changed names when uploading data from Bookshelf to OpenDB.

The fourth problem is to define the appropriate tile size for global and detailed routing. Given a tile size, the die area is divided into tiles to perform global routing, and the global router generates routing guides for detailed routing. The tile size has significant impact on routing congestion and router runtime. Crucially, some academic detailed routers only accept a specific tile size, with tile width/height defined as multiples of routing track pitches. With these tools, the global router must generate routing guides based on a tile size that is acceptable to the detailed router. Therefore, the predefined tile size in past contest benchmarks does not work in certain technology configurations within the academic tool chain. Even worse, the appropriate tile size cannot be calculated with Bookshelf format alone, due to lack of technology information. To address this problem, we define the tile size in the *.route* file as $n\times$ the p^{th} routing layer track pitch, by referring to the technology information in OpenDB. In our present work, we use $n = 15$ and $p = 3$ by default.

The last problem is enabling academic placers to handle *fragmented rows*. Fragmented rows result from undefined or empty placement sites between defined rows in DEF; they are created when commercial tools' macro placement flows intentionally leave spaces between placed macros and standard-cell placement sites. The row fragmentation enhances routability around placed macros, thus increasing chances of DRC-free routing. Since previous Bookshelf benchmarks do not contain any fragmented rows, most Bookshelf-based academic tools do not support fragmented rows. To resolve this problem, we extract undefined or empty placement sites from DEF and add fixed dummy cells to the *.pl* file and *.node* files in our DEF to Bookshelf conversion. The fixed dummy cells function identically as macro halos in preventing standard-cell placements around macros.

[6]In the benchmarks, cells with zero input pins and/or multiple output pins are removed.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MDAT.2022.3179247, IEEE Design and Test



(a) (b) (c) (d) (e) (f)

Fig. 5: Screenshots of routed designs. (a) SkyWater130HD-adaptec1 with Flow1 (routing failed), (b) SkyWater130HD-adaptec1 with Flow2, (c) SkyWater130HD-adaptec1 with Flow3, (d) $12nm$-swerv_wrapper with Flow1, (e) $12nm$-swerv_wrapper with Flow2 and (f) $12nm$-swerv_wrapper with Flow3.

C. Additional Physical Design Features

In the modern physical design flow, inter-stage communications are important to further design optimization. However, with a vertical tool chain such as RDF, where communication is by files, inter-stage communications require upstream information to be transferred along with the flow. In RosettaStone, we enable two additional physical design features for inter-stage optimizations: (i) cell padding and (ii) layer capacity adjustments. Both features are widely used in modern physical design methodology to reduce routing congestion arising in early flow stages, and thus improve routability. RosettaStone's use of OpenDB inherently shares such physical design feature information for inter-stage optimizations.

First, adding cell padding is a methodology that is used to temporarily inflate the size of cells during the global and detailed placement stages to improve routability. This adds additional whitespace at placement to ensure additional space for routers. To support cell padding in Bookshelf, we modify *.nodes* and *.pl* files. We increase standard-cell widths by the user-defined padding width in the *.nodes* file. We also modify the lower-left coordinates in the *.pl* file on each node to match the added padding value. We add padding with the same width on both sides of the cell, so that there is no need to modify the *.nets* file representing the relative coordinates of pins from the center of the node. An experimental case study is described in Section IV-B to demonstrate that cell padding reduces post-route DRC counts.

Second, layer capacity adjustments (i.e., artificial reductions of track supply) are used to prevent the global router from creating routing congestion on lower routing layers. To support layer capacity adjustments, we modify the *.route* file. The *.route* file includes vertical and horizontal routing capacity per tile edge on each layer. We revise the *.route* file by multiplying original capacity by the user-defined percentage reduction of per-layer capacity.

IV. Experimental Results

Our RosettaStone GitHub repository shows integration of 35 open academic benchmarks with four real open-source technologies, demonstrating a range of case studies and academic tools supported by RosettaStone. We have also experimentally confirmed applicability of RosettaStone to closed-source commercial technology.

In this section, we present a sampling of confirming experimental setups and results. We verify our RosettaStone with an open-source technology (SkyWater130HD) and a closed-source commercial technology ($12nm$). The selected academic tools are contest winners and have good performances using Bookshelf or LEF/DEF. We perform placement to routing to show routed designs with various combinations of academic tools. Also, we demonstrate the use of RosettaStone to extend past contest frameworks as well as inter-tool communications for improved flow outcomes.

A. Demonstration of Physical Design Flow with Academic Tools

We first demonstrate integration of past tools and benchmarks with current tools and design enablements via RosettaStone. As shown in Table I, we define three flows with a mix of academic tools. We use two designs, adaptec1 from the ISPD-2005 contest and the RISC-V swerv_wrapper core,[7] with two technologies (SkyWater130HD and foundry $12nm$). We report results for two technology-design pairs (SkyWater130HD-adaptec1 and $12nm$-swerv_wrapper).

Flow1 contains a Bookshelf-based academic tool (NTUplace3) that does not understand $terminal\_NI$ (pre ISPD-2011 contest format). Flow2 contains a Bookshelf-based academic tool (NTUplace4h [4]) that does understand $terminal\_NI$ (post ISPD-2011 contest format,[8]) and Flow3 contains LEF/DEF-based tools in the OpenROAD flow. Figures 5(a)-(f) show routed designs using the three flows; Figure 5(a) shows only a post-placement result since the design is unroutable. In the next subsection, we illustrate how inter-stage communication (use of cell padding) can improve routability.

TABLE I: Academic tools in three flows in the experiments.

Stage	Flow1	Flow2	Flow3
Global Placement	NTUplace3	NTUplace4h	RePlAce
Detailed Placement	NTUplace3	NTUplace4h	OpenDP
Clock Tree Synthesis	TritonCTS	TritonCTS	TritonCTS
Global Routing	FastRoute	FastRoute	FastRoute
Detailed Routing	TritonRoute	TritonRoute	TritonRoute

B. Placement with Padding Awareness

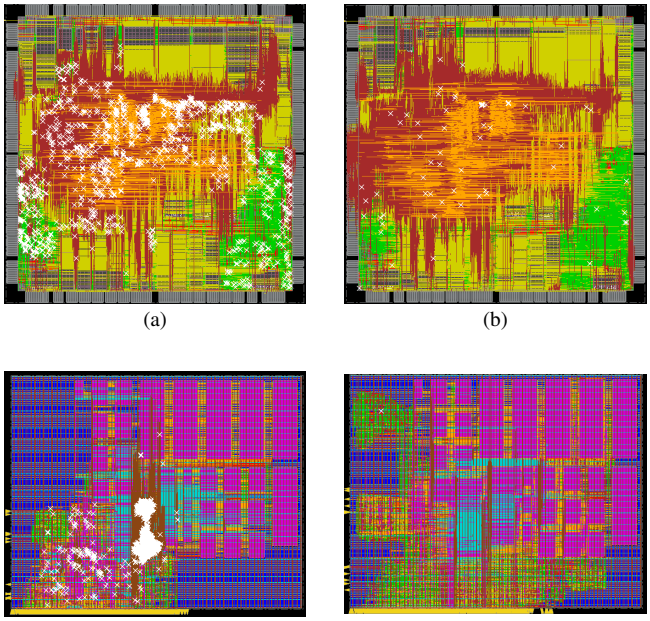
We now show routed results when cell padding is applied at placement to reduce post-route DRC counts. We use the same technology-design pairs (SkyWater130HD-adaptec1 and $12nm$-swerv_wrapper) with Flow1 and Flow2. The tools in Flow3 are in OpenROAD and support cell padding without RosettaStone. We enable two Bookshelf-based placers (NTUplace3 and NTUplace4h) to support cell padding through RosettaStone. Also, we apply cell padding widths from zero to four with a step size of one. Table II gives post-route DRC

[7] https://github.com/westerndigitalcorporation/swerv_eh1

[8] OBS in *.shape* causes problems with the academic placer, so we do not use a *.shape* file for our experiments. Instead, we define the OBS as a fixed dummy cell in the *.pl* file.

TABLE II: Post-route DRC counts and wirelength (WL) with cell padding.

Padding	SkyWater130HD-adaptec1				12nm-swerv_wrapper			
	Flow1		Flow2		Flow1		Flow2	
	#DRC	WL (um)	#DRC	WL (um)	#DRC	WL (um)	#DRC	WL (um)
0	Unroutable	N/A	4259	31056820	10158	1467403	60100	1653790
1	Unroutable	N/A	1552	33133707	39	1461999	3961	1649501
2	Unroutable	N/A	Unroutable	N/A	17	1543437	14	1597491
3	6054	32375129	Unroutable	N/A	6	1601777	4	2025791
4	3490	31411154	112	33104601	3	1676066	1	2067074



(a) (b) (c) (d)

Fig. 6: Screenshots of routed designs with DRC markers. (a) SkyWater130HD-adaptec1 with Flow2 (0-site padding, 4259 DRCs), (b) SkyWater130HD-adaptec1 with Flow2 (4-site padding, 112 DRCs), (c) 12nm-swerv_wrapper with Flow2 (0-site padding, 60100 DRCs) and (d) 12nm-swerv_wrapper with Flow2 (4-site padding, 1 DRC).

results with various cell padding widths.[9] We see that post-route DRC counts decrease when larger cell paddings are added at placement stages in Flow1 and Flow2. For Flow2 and the 12nm-swerv_wrapper, the DRC counts with 0- and 2-site padding are 60100 and 14, respectively; 2-site padding yields 3.4% wirelength reduction compared to 0-site padding. Figures 6(a)-(d) show routed designs with use of cell padding, and white markers that indicate post-route DRC locations. We observe that cell padding significantly reduces DRC markers in Figures 6(b) and 6(d).

C. Timing-aware Integration

We demonstrate timing-aware integration by applying logic path cutting during the benchmark integration, and by applying timing- and electrical rules-driven resizing and buffering during the pre-CTS and post-CTS steps of the OpenROAD flow[10] [1], [9]. We choose SkyWater130HD-adaptec1 design from Table II with Flow1-2 and 4-site padding. We assign the target clock period as 20ns and maxPathLength as 25 for the logic path cutting. Our timing-aware integration results show zero Worst Negative Slack (WNS) after routing, with worst slack values of +5.99ns (Flow1) and +4.25ns (Flow2). Without the resizing/buffering and logic path cutting, less sensible WNS values of -187.01ns (Flow1) and -183.38ns (Flow2) would result.

[9]The two unroutable results in Flow2 are caused by segmentation fault in the global routing tool.

[10]https://github.com/The-OpenROAD-Project/OpenROAD-flow-scripts

V. Conclusion

We have described a new, open-source framework called *RosettaStone* that uses OpenDB to unify academic physical design research and modern industry practice. Our work supports the integration of past academic tools and benchmarks with industry-standard formats and modern testcases for physical design research. RosettaStone enables past and future academic research and contests to have complete flow context and canonical evaluations. In addition, RosettaStone's codes are opened at our GitHub repository, and we share 35 benchmarks integrated into four open-source technologies (ASAP7, NanGate45 and SkyWater130HS/HD).

Acknowledgments

This work is partially supported by DARPA HR0011-18-2-0032 and NSF CCF-2112665.

References

[1] 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", *Proc. DAC*, 2019, pp. 76:1-76:4.

[2] A. E. Caldwell, I. L. Markov and A. B. Kahng, "Toward CAD-IP Reuse: a Web Bookshelf of Fundamental Algorithms", *IEEE Design & Test of Computers*, 19(3) (2002), pp. 70-79.

[3] C.-K. Cheng, A. B. Kahng, I. Kang and L. Wang, "RePlAce: Advancing Solution Quality and Routability Validation in Global Placement", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 38(9) (2018), pp. 1717-1730.

[4] M. K. Hsu, Y. F. Chen, C. C. Huang, S. Chou, T. H. Lin, T. C. Chen and Y. W. Chang, "NTUplace4h: A Novel Routability-Driven Placement Algorithm for Hierarchical Mixed-Size Circuit Designs", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 33(12) (2014), pp. 1914-1927.

[5] T. Chen, Z. Jiang, T. Hsu, H. Chen and Y. Chang, "NTUplace3: An Analytical Placer for Large-Scale Mixed-Size Designs With Preplaced Blocks and Density Constraints", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(7) (2008), pp. 1228-1240.

[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-2020: Strengthening the Foundation for Academic Research in IC Physical Design", *Proc. ICCAD*, 2020, pp. 1-6.

[7] J. Chen, I. H.-R. Jiang, J. Jung, A. B. Kahng, S. Kim, V. N. Kravets, Y.-L. Li, R. Varadarajan and M. Woo "DATC RDF-2021: Design Flow and Beyond", *Proc. ICCAD*, 2021, pp. 1-6.

[8] A. B. Kahng, H. Lee and J. Li, "Horizontal Benchmark Extension for Improved Assessment of Physical CAD Research", *Proc. GLSVLSI*, 2014, pp. 27-32.

[9] A. B. Kahng and T. Spyrou, "The OpenROAD Project: Unleashing Hardware Innovation", *Proc. Government Microcircuit Applications and Critical Technology Conference*, 2021.

[10] W. H. Liu, W. C. Kao, Y. L. Li and K. Y. Chao, "NCTU-GR 2.0: Multithreaded Collision-aware Global Routing with Bounded-Length Maze Routing", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(5) (2013), pp. 709-722.
[11] M. Pan, Y. Xu, Y. Zhiang and C. Chu, "FastRoute: An Efficient and High-Quality Global Router", *VLSI Design* (2012), pp. 1-18. doi:10.1155/2012/608362
[12] T. Spyrou, "OpenDB, OpenROAD's Database", *Workshop on Open-Source EDA Technology*, 2019, pp. 1-2.

Andrew B. Kahng Distinguished professor of CSE and ECE, UC San Diego. Research interests: IC physical design, design-technology co-optimization, machine learning for EDA and IC design, open source, technology roadmapping. Ph.D., Computer Science, UCSD 1989. Professional memberships: IEEE, ACM, SPIE.

UCSD Departments of CSE and ECE, Mailcode #0404, 9500 Gilman Drive, La Jolla, CA 92093-0404.

Phone/fax: 858-822-4884 / 858-534-7029 Email: abk@ucsd.edu

Minsoo Kim Pursuing a Ph.D. degree at UC San Diego, USA. Received the M.S degree at Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 2013. Research interest: Technology-aware physical design methodology, design-technology co-optimization (DTCO) and open-source EDA.

UCSD Departments of CSE and ECE, Mailcode #0404, 9500 Gilman Drive, La Jolla, CA 92093-0404.

Phone: 858-361-6821 Email: mik226@ucsd.edu

Seungwon Kim Postdoctoral Scholar of CSE, UC San Diego, USA. Received the Ph.D. degree at Ulsan National Institute of Science and Technology (UNIST), South Korea in 2019. Research interest: IC physical design flow optimization with machine learning, power delivery optimization, open-source EDA. Professional memberships: IEEE.

UCSD Departments of CSE and ECE, Mailcode #0404, 9500 Gilman Drive, La Jolla, CA 92093-0404.

Phone: 858-642-5334 Email: sek006@ucsd.edu

Mingyu Woo Pursuing a Ph.D. degree at UC San Diego, USA. Received the M.Sc degree at Ulsan National Institute of Science and Technology (UNIST), South Korea in 2018. Research interest: IC physical design, open-source EDA, and machine learning.

UCSD Departments of CSE and ECE, Mailcode #0404, 9500 Gilman Drive, La Jolla, CA 92093-0404.

Phone: 858-900-4742 Email: mwoo@ucsd.edu