DATC RDF: An Open Design Flow from Logic Synthesis to Detailed Routing

Jinwook Jung  
Korea Advanced Institute of Science and Technology  
Daejeon, Korea  
jinwookjung@kaist.ac.kr

Iris Hui-Ru Jiang  
National Taiwan University  
Taipei, Taiwan  
huiru.jiang@gmail.com

Jianli Chen  
Fuzhou University  
Fuzhou, China  
jlchen@fzu.edu.cn

Shih-Ting Lin  
National Chiao Tung University  
Hsinchu, Taiwan  
cxzsd3661512@gmail.com

Yih-Lang Li  
National Chiao Tung University  
Hsinchu, Taiwan  
ylli@cs.nctu.edu.tw

Victor N. Kravets  
IBM T. J. Watson Research Center  
New York, United States  
kravets@us.ibm.com

Gi-Joon Nam  
IBM T. J. Watson Research Center  
New York, United States  
gnam@us.ibm.com

ABSTRACT

In this paper, we present DATC Robust Design Flow (RDF) from logic synthesis to detailed routing. Our goals are 1) to provide an open-source academic design flow from logic synthesis to detailed routing based on existing contest results, 2) to construct a database for design benchmarks and point tool libraries, and 3) to interact with industrial designs by using industrial standard design input/output formats. We also demonstrate RDF in a scalable cloud infrastructure. Design methodology and cross-stage optimization research can be conducted via RDF.

KEYWORDS

VLSI design flow, CAD contest, physical design

1 INTRODUCTION

EDA research contests and their released benchmark suites have successfully attracted research endeavors on timely and practical problems. These contests stimulated innovative solutions which indeed advanced the cutting edge technologies.

Based on these outstanding point tools, the DATC Robust Design Flow (RDF) is developed to provide an open-source academic design flow, which can facilitate design methodology and cross-stage optimization research. Our goals are 1) to provide an academic reference flow from logic synthesis to detailed routing based on existing contest results, 2) to construct a database for design benchmarks and point tool libraries, and 3) to interact with industrial designs by using industrial standard design input/output formats.

2 DATC ROBUST DESIGN FLOW

DATC RDF improves the preliminary versions [11, 12] to deliver a complete research infrastructure of VLSI design flow [7]. Our goal is to provide an open-source academic design flow that covers entire design stages, i.e., from logic synthesis to detailed routing, based on the public academic point tools from the previous EDA research contests [5, 15, 18, 22–24, 27]. Figure 1 illustrates the overview of DATC RDF. It includes academic point tools for logic synthesis, global placement, detailed placement, timing analysis, gate sizing, global routing, and detailed routing. These tools are interfaced via transition scripts that enable data exchange between tools of other domains.

A design library for DATC RDF contains:

- A circuit written in a structural Verilog netlist.
- Standard cell library in Liberty format.
- Physical information of standard cells along with technology information in LEF format, which defines physical dimensions of each cell.
- Initial floorplan described in DEF format.
- Design constraints in SDC (Synopsys Design Constraints) format, such as clock period, driver information of each input port, and load capacitance of each output.

Given a design library, DATC RDF starts with the logic synthesis and generates a logic-optimized gate-mapped Verilog netlist. Taking the netlist and LEF/DEF from the design library, global and detail placements are then performed. Wire parasitics are extracted so that the timing of the placement result can be analyzed. Gate sizing may optionally run to remove timing violations while minimizing leakage power. Legalization is called to remove illegal placement caused by the gate sizing. Finally, global routing and detailed routing is performed.

Currently, RDF database contains:

2. Logic synthesis: ABC.
4. Detailed placers: FastPlace3-DP, MCHL.
5. Global routers: NCTUgr, FastRoute4.1, BFG-R.
8. Timers: OpenTimer, iTimerC2.0.

Users can customize their own flow based on the above options.

3 UPDATES IN THIS VERSION

In this section, we highlight the extension from the preliminary versions [11, 12] of DATC RDF as follows. Details of the logic design flow are presented.
Figure 1: Overview of DATC Robust Design Flow.

3.1 Technology and Standard Cell Libraries
To fully cover the entire VLSI design flow, technology and standard cell libraries have to contain cell timing library (for logic synthesis, gate sizing, and timing analysis), as well as technology information and design rules (for placement and routing). In this regard, two technology libraries are available in the current implementation of RDF. The default technology library is a variant of ISPD 2012/2013 Discrete Gate Sizing Contests [24, 25]. We bring the Liberty standard cell library from the contest library. Since the ISPD 2012/2013 contest benchmark suite does not include a LEF file, which includes technology information and physical dimension of each cell, we take the LEF file generated by the A2A methodology presented in [14].

Another library that DATC RDF supports is the ASAP 7nm library [28, 29]. We take a total of 89 standard cells from the library, which includes basic combinational gates along with some complex gates such as AOI221 or OAI222. Only a basic D-type flip-flop with no reset and set ports is included in our library because the logic synthesis tool [3] incorporated in RDF does not support complex sequencing elements. This library also comes with technology and cell LEF files, which can be readily used for all the placement and routing stages.

3.2 Circuit Netlists
A set of circuit netlists are taken from the TAU 2017 Timing Contest [27] as well as from the IWLS 2005 Benchmarks [1]. They are remapped to the standard cell libraries described in the previous section, and the most critical path delay of each circuit is measured. To set tight timing constraints, the clock period is set to 80% of the critical path delay for each circuit. The number of cells in the netlists range from 352 to 571853.

<table>
<thead>
<tr>
<th>Table 1: Design Rules and Routing Preference Metrics.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design rules</td>
</tr>
<tr>
<td>Open</td>
</tr>
<tr>
<td>Short</td>
</tr>
<tr>
<td>Parallel run length spacing</td>
</tr>
<tr>
<td>End of line spacing</td>
</tr>
</tbody>
</table>

3.3 Detailed Placement
The first EDA research contest, ISPD 2005 contest, focused on mixed-size cell placement [22]. In the past decade, most of research endeavors have been devoted to global placement, and current state-of-the-art placers become mature. Very recently, detailed placement and legalization request novel ideas to handle mixed-cell-height circuits for better power, area, routability, and performance trade-offs. A legalizer removes all cell overlaps, meets complicated design rules and constraints, and preserves the “good” solution provided by global placement as much as possible. Considering the mixed-cell-height standard cell designs with various design rules at advanced technology nodes, 2017 ICCAD held a Mixed-Cell-Height Standard Cell Legalization Contest [5].

In RDF, the recent mixed-cell-height legalizer [31] which won the first place award of the contest is included.

3.4 Detailed Routing
ISPD 2018 Initial Detailed Routing Contest [18] is the first contest that targets detailed routing considering practical design rules and honoring global routing guidance. DATC RDF is extended to accommodate the outcome of the detailed routing contest.

In RDF, global routing and detailed routing read input files based on ISPD 2008 Global Routing Contest [23] and ISPD 2018 Initial Detailed Routing Contest [18], respectively. Since there is no industrial standard format for connecting global routing and detailed routing, we develop a global routing guide translator to translate the output format of ISPD 2008 Global Routing Contest into the input format of routing guide used in ISPD 2018 Initial Detailed Routing Contest. In ISPD 2018 Contest, a group of design rules and routing preference metrics are defined (Table 1) and stored in LEF/DEF files. As in commercial routers, the output of a detailed router follows DEF format that can be read by any commercial layout tools.

Currently, NCTUdr is included, and more tools from winning teams will be included.

4 DATC RDF IN SCALABLE CLOUD INFRASTRUCTURE
Because of the today’s crises of design complexity, quality, and cost, a truly new approach and paradigm of design tools and flows are highly required [6, 13]. To foster such research efforts to the
open-source cloud-based CAD tools based on previous CAD contests, we propose a development flow and an implementation of RDF [8], which can be readily deployed especially in the scalable cloud infrastructure. It consists of three fundamental parts as illustrated in Figure 2: code repositories, continuous integration and containerization, and container orchestration. We expect that they make it easier to collaborate the development of point tools and the design flow, and to deploy the entire system in someone’s own machine or public cloud infrastructure.

Source codes of each point optimization tool are maintained in source code repositories, such as git and Mercurial; RDF itself is also maintained in the code repositories. They are then integrated and containerized into a container image [19], which can be automatically done by continuous integration tools [20]. As the source codes are maintained using source code repositories and continuous integration tools, the implementation of RDF can stay always up to date.

With the containerization, one can readily deploy the entire DATC RDF framework because the container keeps all the necessary libraries and dependencies that are necessary to run RDF. In particular, current mainstream cloud providers, such as Amazon AWS [2], Microsoft Azure [21], and IBM Cloud [10] IBM Cloud, all provide off-the-shelf solutions for automated container deployment engine, e.g., Kubernetes. Besides, scaling the deployment can be easily achieved with the ready-to-use horizontal scaling and load balancing features of container orchestration systems. We also expect that the containerization will realize large-scale parallel architectures of design automation systems.

5 EXPERIMENTS AND DEMONSTRATION
DATC RDF framework is implemented using C++ and Python3. We demonstrate our flow based on a benchmark circuit fft_ispd from the TAU 2017 Timing Contest [27].

The circuit netlist was first unmapped to a generic gate library, and subsequently remapped to our standard cell library using Synopsys DesignCompiler L-2016.03-SP5-5 [26]. It was then synthesized using the ABC logic synthesis and verification platform [3] using the AIG optimization script of Lazy-man synthesis paradigm [30]. Two placement instances were then created using ComPLx [16] and NTUPlace3 [4]. They were then routed with NCTU-GR 2.0 [17] and BFG-R [9] global routers. Finally, detailed routing was performed with NCTUdr. The results are shown in Figure 3, Figure 4, and Figure 5.

6 CONCLUSION
In this paper, we present DATC RDF, which is an open design flow from logic synthesis to detailed routing. We include point tools based on previous EDA research contests and will keep expanding the flow coverage vertically and horizontally. We also demonstrate RDF in cloud infrastructure. RDF can be readily integrated with design methodology and cross-stage optimization research.

ACKNOWLEDGMENT
This work was supported by IEEE CEDA Design Automation Technical Committee (DATC). Special thanks go to tool providers for their generosity.

REFERENCES
Figure 4: Global routing congestion map of fft_ispd. Placement result is obtained using ComPLx placer, and global routing is done by NCTU-GR 2.0. (a) Metal-3, (b) Metal-4, (c) Metal-5 and (d) Metal-6 layers.

Figure 5: Detailed routing results of fft_ispd. Metal-3 to Metal-6 layers are colored with green, yellow, red, and orange, respectively. Placement is done with (a) ComPLx and (b) NTUPlace3.